Jiamu Ni

A Hotkey Interaction Technique that Promotes Hotkeys

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology
Espoo 23.08.2017

Thesis supervisor:

Prof. Antti Oulasvirta

Thesis instructor:

D.Sc Jussi Jokinen

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Jiamu Ni

Title: A Hotkey Interaction Technique that Promotes Hotkeys

Date: 23.8.2017 Language: English Number of pages: 52+8

Department of Communication Ecosystem

Professorship: Communications and Networking Code: S-38

Supervisor: Prof. Antti Oulasvirta

Instructor: D.Sc. Jussi Jokinen

Abstract:
Hotkeys provide fast interactions to support expert performance. Compared to the traditional pointer-based selection of commands, hotkeys have the advantage in reducing task completion time. However, research shows that users have a tendency of favoring menu selections. This is partially caused by how hotkeys are displayed in most linear and toolbar menus. This thesis provides a review of key findings from literature that aim to promote hotkeys. On the base of these findings, this thesis develops design criteria for hotkey displays that promote hotkey use. This thesis also proposes a new interaction technique which displays hotkeys on the keyboard. Finally, a cognitive model is constructed to describe a user's decision-making process of choosing between hotkeys and pointer-based selections when this new hotkey display technique is presented.

Keywords: hotkey, display, retrieval latency, pointer-based menu selection, transition from menu selections to hotkeys

Table of contents

1 Introduction

In computing, hotkeys are widely used to enable efficient interactions. A hotkey, alternatively referred to as a keyboard shortcut, or an accelerator, is a key or a combination of keys which is associated with a specific command or function. It is used to execute the command or function within an application or a system. Therefore, a hotkey provides a shortcut alternative to the pointer-based selection of a command from a linear or toolbar menu. Hotkeys provide a good range of commands with a single key combination and thus possess a global interface scope in which users do not need to go through a menu hierarchy. The users can avoid the action of moving the cursor from the workspace to a graphical widget and then back to the workspace. In many word processing tasks, a user's hands rest on the keyboard. Using hotkeys means the user avoids moving his hand to the mouse and then back to the keyboard after invoking a command. These characteristics enable hotkeys to improve users' productivity, avoid visual distraction from users' current tasks to GUIs, and provide precision in many tasks such as text editing. A variety of empirical studies [e.g. 1, 2] demonstrate these advantages of hotkeys.

However, despite their potential of facilitating the transition from novice to expert performance, hotkeys are underused and most users still prefer pointer-based selections of commands on Graphical User Interfaces (GUIs). Carroll and Rosson's "paradox of the active user" [3] suggests two biases to explain this user tendency: a production bias, which encourages users to reuse the existing knowledge rather than discovering new strategies which will eventually improve the user performance; and an assimilation bias, which encourages users to apply the existing knowledge to interpret and accomplish new tasks. Both biases encourage users to continue with pointer-based menu selections that they are familiar with and create an associated user tendency of skipping the opportunity to learn hotkeys and improve the user performance. Additionally, Fu and Gray [4] explain that if a suboptimal strategy is sufficiently practiced, applicable to most tasks, and able to provide fast feedback, then it is preferred for interaction over the optimal alternative strategy. This explains the success of the conventional design of GUIs that enable users to quickly learn how to find and invoke the commands with the cursor.

The low user acceptance of hotkeys draws attention to an important issue concerning usability: although the advantage of using hotkeys in certain actions as regards task completion time can be small, the performance difference between hotkeys and pointer-based menu selections will increase when selections of certain commands require a user to navigate through the menu hierarchy (e.g. a cascade menu). Moreover, the small performance difference can accumulate if the users' tasks involve daily repetitions of selection from a set of commands. Due to the low popularity of using hotkeys, there has been an increase in literature specifically addressing the problem of how user interfaces should be designed to facilitate the transition from the pointer-based menu selections to hotkeys. Several studies have specified that the hotkey interaction techniques implemented in most menu systems are the major cause for hotkeys being underused

[see e.g. 5, 6]. It is, therefore, necessary to identify the limitations of these hotkey interaction techniques and establish design principles of hotkey interactions which can encourage users to adopt the use of hotkeys.

This thesis develops a hotkey interaction technique that uses a new approach for displaying hotkeys. This new hotkey display is designed to facilitate the transition from pointer-based menu selections to hotkeys. It provides a review and analysis of different hotkey interaction techniques that are designed with the purpose of promoting hotkey use. On the basis of these techniques, it summarizes the existing knowledge about designing hotkey interactions, while highlighting successful approaches that facilitate hotkey adoption in order to provide a foundation for underlying design principles. These principles are applied to the set of design variations developed in this thesis to select and validate the optimal design.

This thesis also constructs a model for a user's decision-making process when choosing between hotkeys and pointer-based menu selections. After a hotkey has been displayed, a user encodes the command-hotkey paired association in his memory. Using the encoded hotkey is associated with a mental preparation stage where a user tries to retrieve it from his Long Term Memory (LTM). The hotkey retrieval latency and the probability of a successful retrieval determine whether the user will choose the hotkey to accomplish a task. The hotkey retrievals add a heavy load on the user's memory. However, there are no detailed studies on how hotkeys are encoded and how their retrieval latencies are calculated. In this thesis, an equation is constructed to predict the retrieval latency of a hotkey when a user is trying to retrieve multiple hotkeys in a row. With the help of this equation, it becomes feasible to model a user's decision-making process of choosing between hotkey and menu selections.

1.1 See and Point *vs.* Learn and Remember

GUIs such as linear and toolbar menus have been the primary interfaces that mediate between users and computer applications. Their success is mainly brought by their support for novice users [10]. The characteristics of menu selections are indicated by the phrase "see and point" [7] since novice users are benefited from the direct manipulation principle followed by GUIs. Users are able to visually search salient graphical widgets organized into labeled menus and manipulate them through pointing with a cursor, a stylus or a finger. Since pointer-based menus place the knowledge in the environment that users interact with, they rely on knowledge-in-the-world. Such knowledge is embedded in the activities and processes which users are engaged in and is not an attribute possessed by users [8]. Pointer-based menu selections are interaction-intensive and effective with novice users.

Although direct manipulation-based GUIs are easy to understand and learn, it constrains the full use of users' memory capacities and limits the performance of expert users by compelling them to work with relatively slow actions. Conversely, hotkeys are specifically designed for supporting expert performance. The characteristics of hotkey use can be described by the phrase "learn and remember" [7] since using hotkeys requires users to learn and memorize the command-hotkey paired associations

beforehand. Therefore, using hotkeys greatly relies on knowledge-in-the-head which is a memory-intensive strategy and requires extensive practice.

While hotkeys are effective to expert users, they exhibit three major limitations:

- Hotkeys are difficult to learn: learning and memorizing the association between a command and its corresponding hotkey place high demands on a user's Long Term Memory, especially when the associations are arbitrarily assigned. As the number of commands increase, the amount of meaningful assignments quickly run out, and arbitrary assignments are inevitable. For example, it is quite easy to associate the command "Copy" with its corresponding hotkey "Ctrl + C" since the letter C is the initial letter of the command. However, for the command "Cut", the corresponding hotkey "Ctrl + X" is arbitrarily assigned and is more difficult to recall.

- Hotkeys lack in consistency: hotkeys are inconsistent across applications. For example, the hotkey "Ctrl + H" corresponds to the command "Search and Replace" in Microsoft Word, but it corresponds to the command "Open History" in the Firefox browser. This lack of consistency increases the difficulty for learning hotkeys.

- Hotkeys lack visibility: commonly, hotkey displays are accessed through menus in which their associated commands are also presented. Therefore, to find a hotkey, a user has to first locate its corresponding command. In some extreme cases, hotkeys are only displayed through documentation. For example, the search function in the Windows operation system is frequently used for searching files, applications and settings. Its corresponding hotkey "Windows + S" can only be found in the documentation. Such interactions deter users from adopting hotkeys.

This thesis focuses on alleviating the problem associated with the lack of visibility of hotkeys by designing a new hotkey display technique which increases the exposure of hotkeys and draws users' attention to the hotkey display.

From the perspective of hotkey learning, meaningful assignments for command-hotkey paired associations, such as "Ctrl + C" for "Copy", are easier for users to recall than arbitrary non-meaningful assignments such as "Ctrl + X" for "Cut". Shneiderman distinguishes between "meaningful apprehension of relations" and "senseless drill and arbitrary associations" [15]. According to Shneiderman [15], non-meaningful assignments belong to syntactic knowledge that is acquired through rote learning. They do not integrate well within existing systems of semantic knowledge. For example, the hotkey "Ctrl + X" corresponds to the command "Cut". But the letter "X" is a symbol which does not directly correspond to "Cut". Users need to repeat "Ctrl + X is Cut" to memorize the assignment. Contrarily, meaningful assignments belong to semantic knowledge that is acquired through meaningful learning which encourages users to assimilate new hotkeys within existing semantic knowledge. For instance, when a user encounters a new hotkey "Ctrl + C", he associates letter C with the initial letter of the command "Copy". Thus, letter C is not a mere symbol but possesses a meaning. The results of Shneiderman's experiment suggest that meaningful assignments are easier to

comprehend since the letter key is conveyed by the command's initial letter [15]. Meaningful assignments for command-hotkey paired associations are beyond the scope of this thesis which focuses on the hotkey interaction techniques.

1.2 Two Most Popular Hotkey Interaction Techniques

Some hotkey display techniques that aim to improve the visibility of hotkeys have been implemented to different applications. One technique is to display hotkey labels on the left side of the corresponding textual commands in a dropdown menu, as depicted in Figure 1.

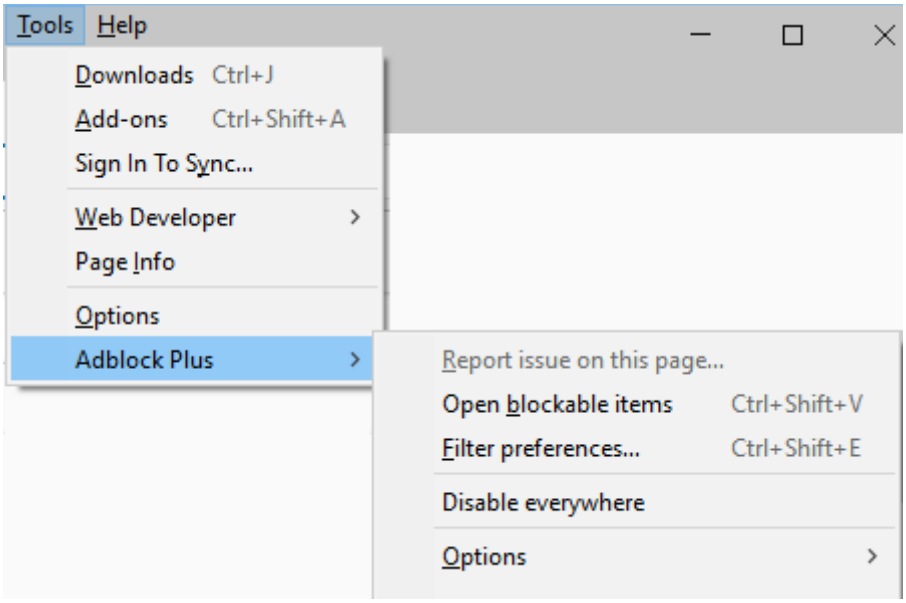

Figure 1: Hotkeys displayed in a dropdown menu

The other technique is designed for displaying hotkeys in a toolbar menu. A hotkey is displayed when a user moves the cursor on its corresponding command which is in the form of a graphical widget, as exemplified in Figure 2.

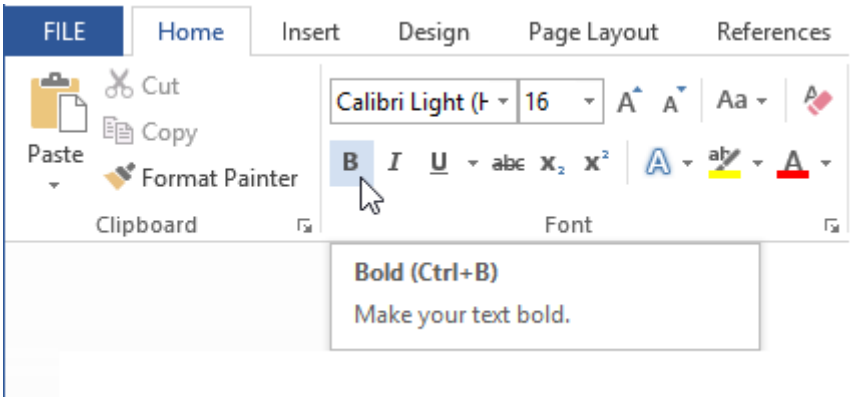

Figure 2: A hotkey displayed in a toolbar menu.

These two hotkey interaction techniques share many similarities. Both require a user to take a pointer-based approach to access the hotkey display. Hotkeys are not displayed until a user actively posts a menu command or hovers the cursor on a toolbar graphical widget. Consequently, displaying hotkeys involves pointing and clicking with the cursor and hotkeys are displayed at the stage in which one more click would be able to accomplish the task. Instead of a simple click, a user needs to pause, wait for the hotkey feedback, and encode the hotkey before invoking it. These actions introduce a performance dip which discourages users from adopting hotkeys and traps users in the novice mode of pointing and clicking [9]. The performance dip is depicted in Figure 3.

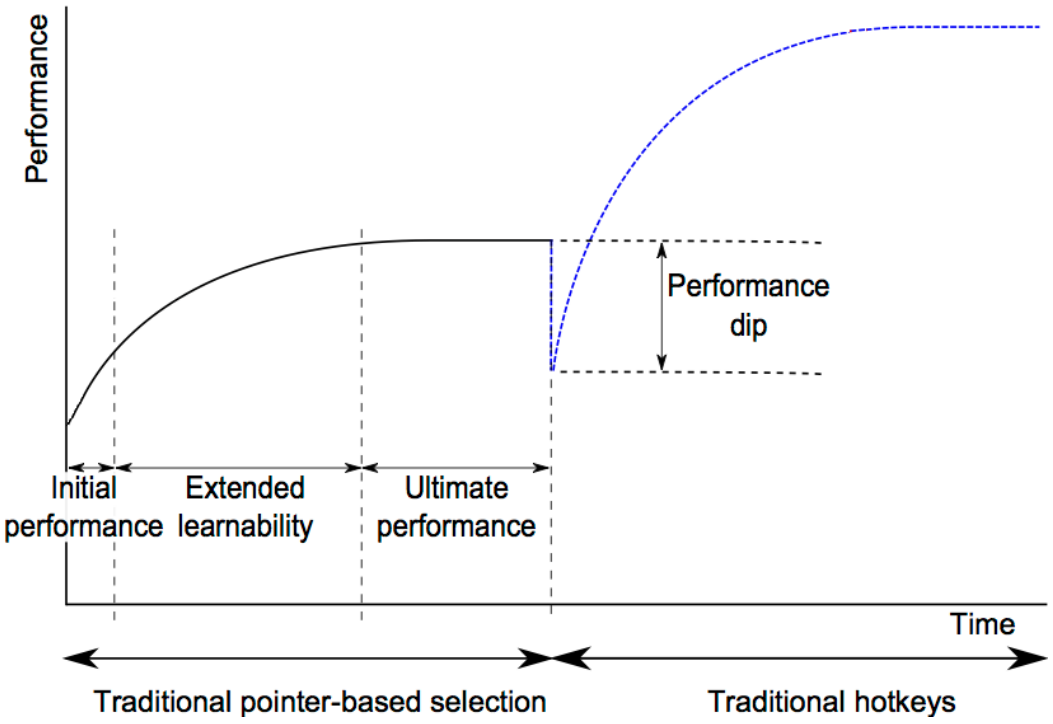

Figure 3: Performance dip induced by traditional hotkey display [9].

As indicated in Figure 3, a performance dip is induced by the transition from the pointer-based menu selections to the hotkeys through traditional hotkey displays. The extent of the dip is influenced by the magnitude of the difference between the techniques used in these two interfaces [10]. Therefore, a user's motivation of learning to use hotkeys is greatly lessened by accessing the hotkey display through a non-hotkey modality such as a cursor. For this reason, these two widely implemented hotkey interaction techniques fail to promote hotkey use.

1.3 Supporting the Initial Transition

To support the transition from pointer-based menu selections to hotkeys, the first stage is to help novice users be aware of the new alternative. This support can be provided through a broad range of approaches that vary in effectiveness and impact. For example,

the previously introduced hotkey display techniques which have been implemented in most dropdown and toolbar menus display hotkeys through pointer-based access. It is unlikely that users would pay attention to the hotkeys during their interactions with the menus. Such techniques are ineffective and fail to draw the users' attention to the existence of hotkeys.

A hotkey interaction technique can implement forceful awareness mechanisms which require users to experience hotkeys by demanding that actions are completed after users practice the use of hotkeys [11]. These mechanisms include audio and visual schemes which expose users to hotkeys. A hotkey interaction technique which applies a forceful awareness mechanism does not require users to take any action to access the hotkey display. When a user completes a task through a pointer-based menu selection, the command's corresponding hotkey is automatically displayed visually or audibly as feedback. For example, after a user clicks on a command in a dropdown menu, its corresponding hotkey is displayed vocally. A visual approach can display the hotkey in textual form which remains on the computer screen until the user presses the hotkey's corresponding keys. A less forceful approach to awareness is to present the hotkey in textual form within a transparent window which slides down through the computer screen's corners. This approach makes the information of hotkeys available, but without demanding the users' attention.

Kurtenbach et al. [11] suggest that after a user is aware of the existence of hotkeys, the probability of the transition from pointer-based menu selections to hotkeys is affected by the user's perception of the future efficiency of using hotkeys. Therefore, the second stage of supporting the transition is to help users be aware of the performance improvement brought on by the use of hotkeys. Malacria et al. [12] propose the implementation of interactive components called 'skillmeters' which indicate the most recent task completion times and the strategies applied to the tasks, as well as the prediction of the task completion times if the alternative strategies would have been used. Thus, users can observe the performance difference between hotkeys and pointer-based menu selections and be motivated to make the transition to hotkey use. However, the prediction of task completion time using hotkeys is based on the Keystroke Level Model (KLM) [13] which is not able to accurately predict the hotkey retrieval latency. A task completion time for using a hotkey is the sum of the motor action of key pressing and the hotkey's retrieval latency which decreases through practice. When a user begins to use hotkeys, the retrieval latency of a hotkey can be high and the performance difference between hotkeys and pointer-based menu selections is minimal. There is a possibility that exaggerated benefits of adapting hotkeys lead to distrust and users will discontinue with the program [14].

Even if users are able to correctly predict the benefits of adopting hotkeys, they might still continue with pointer-based menu selections which they are familiar with [3]. To comprehend this issue, Fu and Gray [4] constructed cognitive models of different tasks and collected data traces of the tasks' actual execution. Their findings suggest that users prefer strategies that are (1) well-practiced and generic and (2) composed of interactive components that are fast and incremental [4]. These two biases greatly reduce the influence of the awareness of performance difference. Therefore, instead of making users aware of the efficiency of using hotkeys, the second stage of supporting the initial

transition is to encourage users to practice hotkey use and lessen their reliance on pointer-based menu selections. Practicing hotkeys shortens the retrieval latencies of hotkeys and thus, increases users' performance in task completion times. Eventually users will perceive the benefits of adopting hotkey use.

1.4 The Rehearsal Design Guideline

In order to facilitate the transition from pointer-based menu selections to hotkeys, the interface must implement a technique that provides support for a novice user to become an expert. According to Kurtenbach [11], pointer-based menu selections form the novice component which allows users to visually search and recognize the commands, and invoke them with the cursor. The transition component consists of the hotkey interaction technique which enables users to switch between menu selections and hotkeys to practice and learn the command-hotkey paired associations. The hotkeys constitute the expert component that supports users' recollection of a command's corresponding hotkey from the Long Term Memory (LTM). These three components are presented in Figure 4:

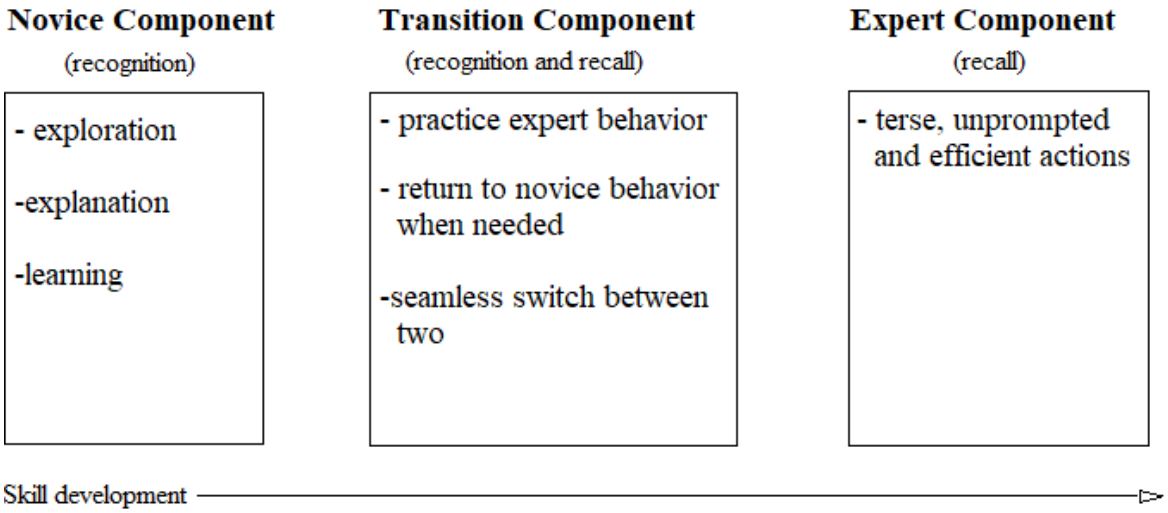

Figure 4: The components for the transition from pointer-based menu selections to hotkeys [11].

Figure 4 presents the three components and the functions of each component. Since the hotkey interaction technique acts as the transition component, and also with the consideration of providing support for the initial transition, the design criteria for the technique should contain the following requirements:

- The interaction technique makes users aware of the existence of hotkeys.
- It encourages users to practice hotkey use.
- It allows users to easily relearn a hotkey if they fail to retrieve it from their Long Term Memory.
- It provides a seamless switch between pointer-based selections and hotkeys.

1.5 The Scope of This Thesis

The goal of promoting hotkey use can be achieved through many different methods. For example, from the perspective of meaningful assignments of command-hotkey paired associations, this goal can be achieved through applying rule-based command-hotkey assignments (e.g. 'Ctrl + O' corresponds to the command 'Open') which to some extent, is easier for users to recall.

This thesis aims to promote hotkey use from the angle of how users interact with the hotkey display. Until now the discussion has covered the initial transition from pointer-based selections to hotkeys and revealed three properties which should be included in an interaction technique designed to promote hotkeys. In the following chapter, alternative hotkey interaction techniques and related works are evaluated to complete the design criteria that are used in this thesis as design guidelines. Finally, this thesis covers two major aspects:

(1) A new hotkey interaction technique which follows the design criteria and facilitates the transition from pointer-based menu selections to hotkeys.

(2) A cognitive model that explains a user's decision-making process of choosing between hotkeys and pointer-based selections when the interaction technique is used in the system.

2 Literature Review

The traditional approaches for hotkey display are pointer-based and bring a performance dip to users who are intended to adopt hotkey use. For example, in a linear menu, hotkeys are placed next to their corresponding commands. A user must first access the menu and locate the command, after which he can find the corresponding hotkey. At this stage, menu selection is almost completed. Therefore, it is unlikely that the user will exit the menu and accomplish the task with the hotkey.

In order to promote hotkey use, many researches have been contemplating on creating new hotkey display techniques that avoid any pointer-based access to hotkey displays. This chapter will introduce three alternative hotkey display techniques and analyze their successes and limitations. The analysis is then used to acquire design criteria for selecting the optimal hotkey display technique from the design variations that are developed in this thesis.

2.1 Analysis of Alternative Hotkey Display Techniques

There are two main strategies that can be used in hotkey display techniques to promote hotkey use: (1) manipulating the menu feedback, which increases the visibility of command-hotkey paired associations to catch users' attentions when they select or browse commands, and (2) manipulating the menu cost, which increases the difficulty or time cost of using pointer-based menu selections [17]. These strategies support both intentional and incidental learning. Increasing the visibility of hotkeys can intensify the exposure to command-hotkey stimuli and facilitate incidental learning of hotkeys. Increasing the menu cost provides users with the incentive to learn hotkeys and facilitates intentional learning of hotkeys. The following section introduces three hotkey display techniques that employ these strategies.

ExposeHotKey$_T$

This simple interactive display technique is a variety of ExposeHotKey [6] and manipulates the menu feedback. It is designed to promote hotkey use in toolbar menus. Displaying a hotkey in a toolbar menu normally requires a user to move the cursor onto the graphical widget of a command. But once the user has moved the cursor onto the widget, the task can be accomplished with only one click. Such hotkey display deters users from hotkey learning and use. To facilitate hotkey use, ExposeHotKey$_T$ displays all the widgets' corresponding hotkeys in a toolbar menu at once. A user accesses the hotkey display through pressing a modifier key (e.g. Command). When the modifier key is held down, the hotkeys overlay under the corresponding graphical widgets as shown in Image 1.

Pros: ExposeHotKey follows Kurtenbach's rehearsal design principle [16] which suggests that accessing the hotkey display should allow users to rehearse the act of

using hotkeys. Therefore, ExposeHotKey$_T$ displays all hotkeys in a toolbar menu without engaging any non-hotkey modality (e.g. the cursor). Instead of accessing the hotkey display by moving the cursor on top of a widget, a user activates the hotkey display by holding down a modifier key, which enables a user to reveal the hotkey display with solely the hotkey modality [6].

Using a single hotkey modality avoids pointer-based menu selection and thus avoids the performance dip induces by it. Besides displaying hotkeys, pressing the modifier key also initiates the action of using a hotkey to accomplish a task. This hotkey display technique has a wide compatibility with other traditional interfaces such as linear menu and ribbon interfaces.

Image 1: The ExposeHotkey$_T$ display in a toolbar [6].

Cons: ExposeHotKey$_T$ requires explicit instructions for how to display hotkeys for users who are not familiar with the system. This requirement raises a question of how a user can discover the mechanism of pressing a modifier key to access hotkey display. ExposeHotKey$_T$ does not provide any visual presentation to interact with users until a user accidently presses the modifier key. Therefore, the lack of implicit instruction weakens the affordance of ExposeHotKey.

Another question is whether ExposeHotKey$_T$ can motivate users who are accompanied with pointer-based menu selections to break their inertia in continuing with the mouse. In the pilot studies, some subjects who were instructed to use hotkeys still preferred to accomplish the tasks with pointer-based menu selections until they were given verbal instructions to change to hotkeys [6]. Breaking such inertia via ExposeHotKey$_T$ is very challenging.

ExposeHotKey$_T$ is an effective approach for simple toolbar menus which have few graphical widgets to be displayed at once. Complicated toolbar menus can contain several levels of menu cascades. ExposeHotKey$_T$ does not display the hotkeys which correspond to widgets that are on the second level onwards menu cascades. Therefore, some degree of pointer-based selection is still inevitable [6].

Audio Feedback

This is a feedback-based approach which provides users with auditory feedback upon pointer-based menu selections to promote hotkey use. After a user accomplishes a task via menu selection, there is a vocal instruction which states the used command and its corresponding hotkey. This approach keeps the hotkey in context in the echoic memory [17].

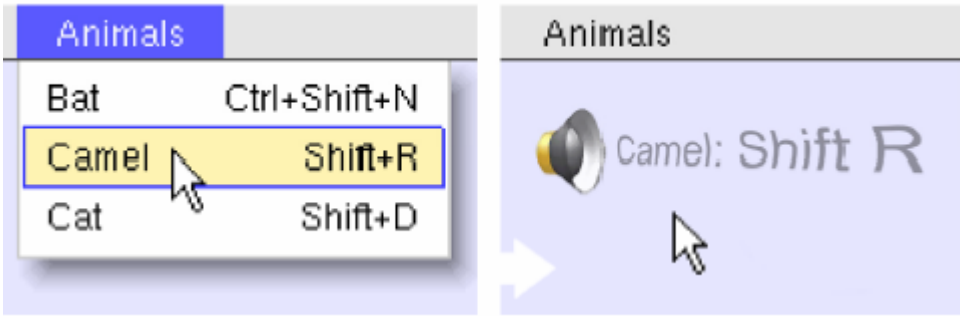

Image 2: Auditory feedback of a command and its corresponding hotkey [17].

Pros: This approach increases the visibility and persistence of the command-hotkey paired associations without changing the traditional menu interfaces, which makes it highly compatible to any interface. The user is not required to take any action to access the hotkey display. Hence, revealing the hotkey display mechanism is not dependent on a new user's unintended action. The auditory feedback follows the on-line approach for teaching hotkeys, which emphasizes that users should learn the hotkeys within the flow of their primary tasks [17]. Auditory display also avoids adding more visual loads to the already congested visual channels.

Cons: Audio feedback has been proven to be one of the most effective ways to promote hotkey use [17]. However, the results are only supported by laboratory experiments in which the disadvantages of auditory display are conveniently neglected.

Auditory feedback of hotkeys requires extra sound equipment such as speakers and headphones which might not always be available. Compared to visual display, auditory display is much more intrusive. Not only does the user hear the auditory feedback but so do the people around him, which causes disturbance to others unless the user always wears a headset. The effectiveness of auditory feedback also depends on a user's working environment. If a user works in a noisy environment, the auditory feedback will lose its meaning. Unlike a visual display which remains until a user exits from it, auditory feedback quickly vanishes. If a user does not encode it the first time, he will have to repeat the process of activating the auditory feedback.

System Delay

This is a cost-based approach which introduces a delay to the pointer-based menu selections. After a user selects a command and exits the menu, there is a two second-delay before the command is put into effect. During the delay, the cursor disappears. A green bar appears on the screen to indicate the progress and the command's

corresponding hotkey is displayed next to the green bar as depicted in Image 3. After two seconds, the green bar disappears and the command is implemented [17].

Pros: This display technique follows a cost-based approach which increases the time cost of pointer-based menu selections to promote hotkey use. The system delay increases the visibility of the hotkeys every time when commands are selected. Hiding the cursor is also an effective approach to attract users' attentions [17]. For users who intend to learn hotkeys, it provides them with the incentive to use hotkeys to avoid the two-second system delay. For users who unintentionally access the hotkey display, it provides a visual stimulus which serves as a long exposure to the hotkey.

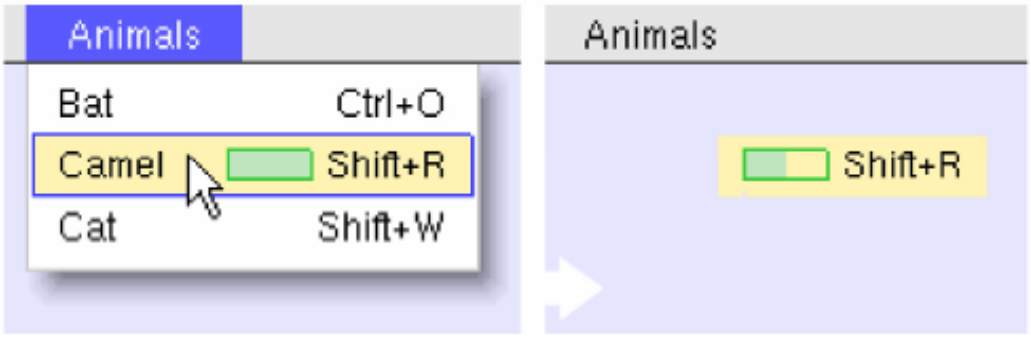

Image 3: A system delay is used on the pointer-based menu selections. The green bar is a time counter that indicates the progress [17].

Cons: The two-second system delay can be negatively perceived by the users and regarded as frustration. During this time, a user is supposed to attend to the hotkey display. However, the experiment's result shows that the users ignored the visual stimulus [17] during the two second-delay. Therefore, displaying hotkeys together with a green progress bar is not a strong enough visual stimulus. This approach creates a two second idle period during which a user cannot do anything since the cursor has disappeared. This cost is perceived by users as a nuisance and fails to motivate users to adopt hotkey use.

2.2 Lessons from the Alternative Hotkey Displays

The previous section introduced three hotkey display techniques which provide valuable insight into the design principles to promote hotkey use. Some of the principles are outlined below.

Exposure: The more often a hotkey is displayed and presented to a user, the more it is retained in the user's memory [17]. Since most frequently used hotkeys have higher learning values than the rest, they should be displayed more frequently. The exposure time of the hotkey should be long enough so that the user has enough time to complete the cognitive processing.

Attention: A user's attention plays a crucial role in the hotkey learning process [17, 18]. Therefore, a hotkey display technique should be able to bring a user's attention to the

command-hotkey paired associations when the user invokes a command via pointer-based menu selection.

Modality for accessing hotkey display: Research has shown that accessing the hotkey display through the pointer-based modality introduces a performance dip [6] and fails to promote hotkey use. Therefore, the hotkey display should be accessed without any pointing action to avoid non-hotkey modality.

Incidence learning: Learning hotkeys does not necessarily have to be effortful. Instead, hotkey learning can take place when a user is exposed to a visual stimulus which displays a hotkey. This approach solves the problem of how a user can discover the access to the hotkey display. Moreover, rather than choosing hotkeys, users tend to prefer pointer-based menu selections which they have learned first. This inertia reduces the likelihood for novice users to actively display and learn hotkeys. Therefore, hotkey display should be a by-product of pointer-based menu selections and be displayed automatically. A hotkey display technique which provides incidental learning can give users the initial motivation to learn hotkeys.

2.3 Other Related Work

There is intensive literature on the principles and guidelines for designing display technologies that promote hotkey uses. This includes strategies supporting novice to expert transitions in user interfaces [10], the theoretical accounts for keystroke-driven user interfaces (e.g. hotkeys) and pointer-based menu-driven user interfaces (e.g. menu selection) in terms of learning, forgetting and relearning [19]. This also includes Soft Constraint Hypothesis (HCS) which specifies that an information system incorporating cognition with perceptual-motor operations preserves the resource of time instead of cognition [20].

The prior work which specifically focuses on hotkeys includes the performance comparisons between shortcut techniques (e.g. hotkeys, stroke gesture) [21], and a specially designed touch-sensitive keypad which detects which hotkey will be pressed and provides a preview of the result of the key pressing [22]. The work also includes the HKC tool designed to help users exceed the visual characteristics to learn hotkeys [23]. Additionally, one study demonstrates the significance of automatic display of commands' corresponding hotkeys to support incidental learning of hotkeys [24]. Finally, literature on the Adaptive Control of Thought-Rational (ACT-R) theory provides computational models developed within cognitive architectures to predict single and multiple hotkey(s) retrieval latencies for users with different working memory capacities [25, 26, 27].

2.4 Design Criteria for Hotkey Display Techniques

The previous chapter developed a part of the design criteria by considering the importance of providing supports for the initial transition and the rehearsal design guideline. In this chapter, three alternative hotkey display techniques are analyzed to complete the design criteria which are listed as following:

The hotkey display technique

- makes users aware of the existence of hotkeys
- encourages users to practice hotkey use
- allows users to easily relearn a hotkey if they fail to retrieve it from their Long Term Memory
- provides a seamless switch between pointer-based selections and hotkeys
- provides a long enough exposure of the hotkeys so that users can encode them
- catches users' attention
- avoids the use of any pointer-based modality to access the hotkey display
- supports incidental learning

In the next chapter, a variety of hotkey display techniques are presented. The design criteria are used to select the optimal hotkey display technique.

3 Variations of Hotkey Display Techniques

A user interface (UI) can be defined according to different perspectives. Symonds and Khosrow-Pour define it as “the aggregate of means by which the users interact with a particular machine, device, computer program or other complex tool” [28]. Additionally, Oulasvirta defines a user interface as “a system by which the users interact with a computing technology that has more than one internal state and thereby its purpose is to assist the users to accomplish their goals [29]”. Also, according to Myers, a user interface of a computer program is the part that handles the output to the display and the input from the person [30]. These definitions present the following features of a user interface.

- A UI provides interaction between the users and the system.
- It allows the users to manipulate the system through input.
- It allows the system to produce a response to the manipulation and display the response through output.
- It helps the users to complete their tasks.

Therefore, the design of a UI determines how the system’s output is displayed and how the users interact with the interface. A good UI design enables the users to accomplish goals with high accuracy and efficiency. Moreover, it should be intuitive and easily accessed by the users. In order to achieve these goals, the design of a UI must consider the users’ cognitive and emotional perspectives [31]. An interesting example which shows the importance of the psychological perspective of a UI is the piano stair in a subway entrance in Stockholm. The picture of the piano staircase is presented in Image 4 [32].

Image 4: The piano staircase at the Stockholm subway entrance.

As shown in Image 4, the steps in the staircase are modified according to the layout of a piano keyboard. When a user walks on the steps, he will hear the sound of the piano keys. The outcome of the modification is that most users prefer the piano staircase to the escalator because the unusual steps generate the emotion of surprise, which, in turn creates a sense of curiosity in the users and motivates them to try the staircase. Although from the perspective of usability, escalators are more efficient and require less physical effort, the users' selection between these two UIs are greatly affected by their emotions. Therefore, designing UI has become a combination of information technology and psychological science.

This chapter presents nine alternative hotkey display techniques that potentially promote hotkey use. These display techniques are compared on the basis of the modalities of display (visual vs auditory), the devices for display (keyboard-based display vs screen-based display), and the design criteria that are developed in the previous chapter.

3.1 Design Questions for Hotkey Display Techniques

A hotkey display technique is the means used to present the hotkeys to the users. It determines how the users access the hotkeys. In this thesis, the designs of the hotkey display techniques are guided by the mind map which describes the interaction between the users and the computer systems when they access the hotkey display.

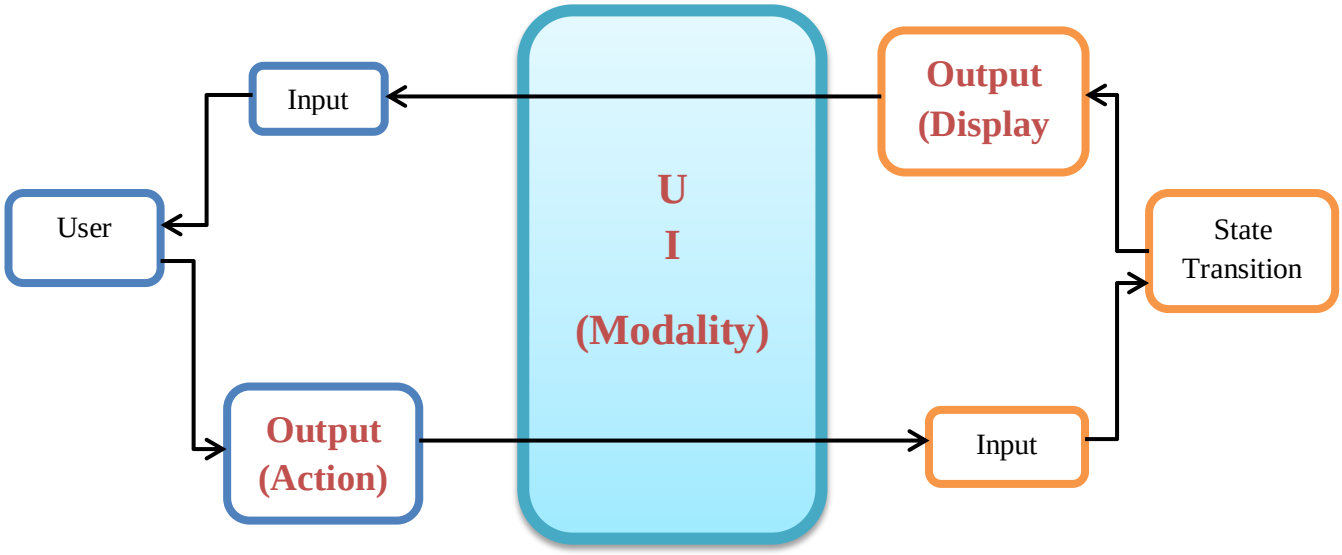

Figure 5: A mind map of the user-hotkey interaction process.

As depicted in the mind map in Figure 5, the users take actions to activate the hotkey display (e.g. press a modifier key or click on the hotkey list with a pointer). Their actions are interpreted as inputs and transferred through the user interface to the computer system. Then, the inputs generate state transitions in the system, which produces hotkey display as outputs. Finally, the hotkey display is perceived by the users

as input information to enable them to complete tasks with hotkeys. The interaction process shows that the design of a hotkey display is determined by three key components of the human-computer interaction (marked in red in the mind map): the user interface, the system's output (i.e. hotkey display), and the user's output (i.e. their actions that activate the hotkey display). Each component is associated with a design question and by finding solutions for these questions different hotkey display techniques are developed. The design questions and their solutions are listed below.

Component	Design Question	Solutions
UI	Which modality does the UI take?	Visual and auditory
System's output	Where are the hotkeys displayed?	On the screen or on the keyboard
Users' output	What actions do users take to activate the hotkey display?	Pointer-based selection, automatic display, and hotkey-based access

Table 1: A hotkey display's key components, related design questions and possible solutions.

Screen-based visual hotkey display techniques

1. Hotkey List

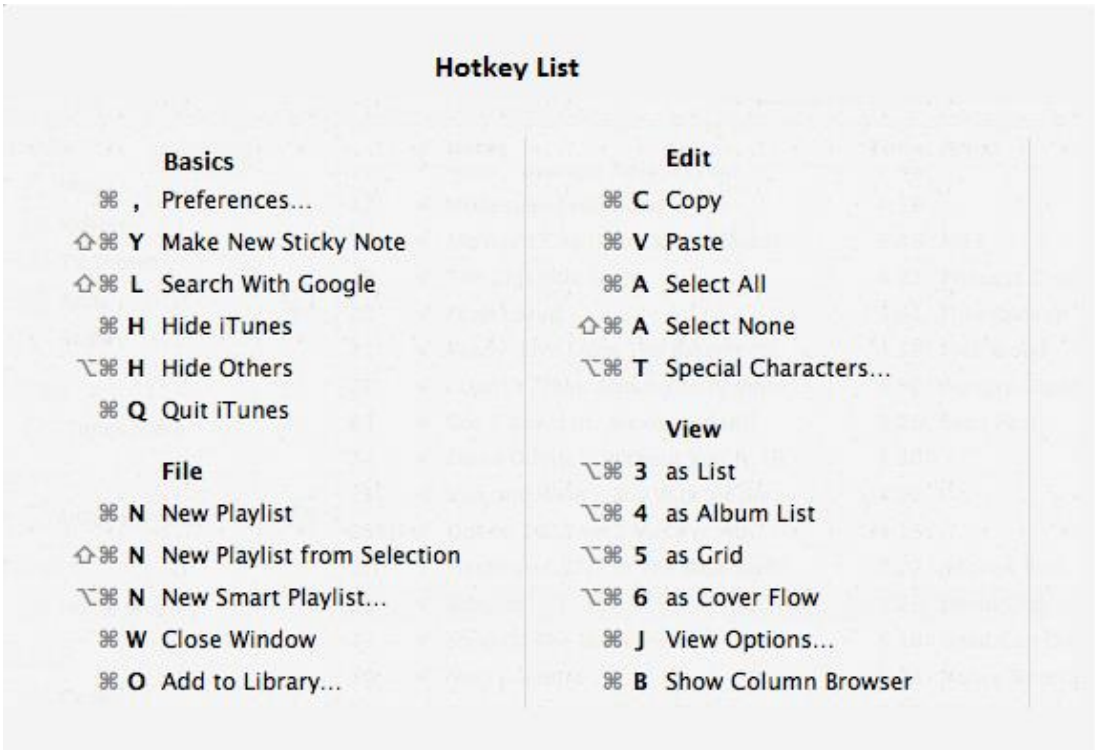

Image 5: A hotkey list for iOS system

Hotkey list is a screen-based display technique. Hotkeys are displayed on a list and a user activates the hotkey list by pressing a modifier key (e.g. Command, Ctrl). The hotkeys can be grouped according to their alphabetic order or be categorized according

to their functionalities. In order to make the search of the most regular hotkeys fast, six basic hotkeys are displayed on the top of the hotkey list. This list disappears after the user releases the modifier key. In Image 5, an example of hotkey list is shown.

2. Search Panel

Computer applications often have tens of hotkeys. For instance, Microsoft word 2016 has 93 hotkeys [33]. Creating a hotkey list for such applications may not be feasible since it takes too much time for the users to search for a particular hotkey in the list. A search panel also saves time. By pressing a modifier key, a user activates the search panel on the screen and then he types the command in the panel and the corresponding hotkey is displayed as the search result. By pressing the "Esc" button, the user deactivates the search panel. Image 6 presents an example of the search panel for hotkeys.

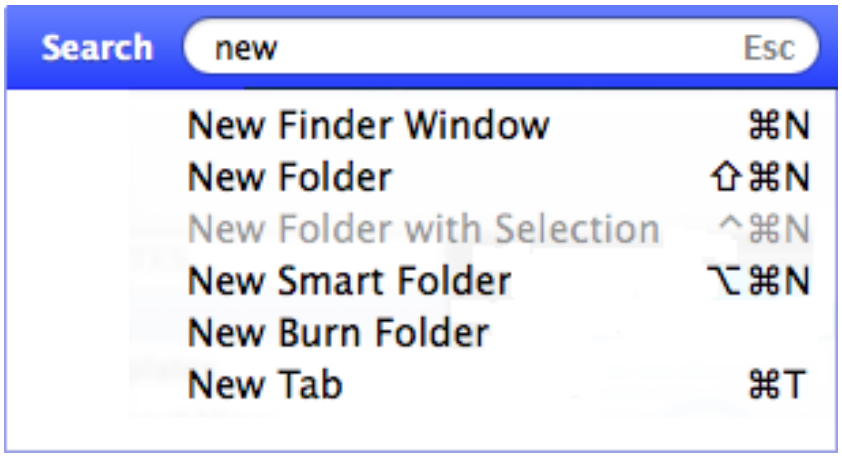

Image 6: Search panel for hotkeys

3. Hotkey List with a Search Panel

The search panel and the hotkey list can be combined to create more varieties. For example, a list of nine previously searched hotkeys can be placed above the search panel. If a user can find the target hotkey in the list, he can save the time spent typing the hotkey's corresponding menu item into the search panel.

4. Alphabetical Hotkey List

The hotkeys are grouped in an alphabetical order and the hotkey list displays only the hotkeys with the same initial letter. The modifier key "alt" is used to access the list and Command is used for the operation. The users access the list by pressing alt + initial letter. For example, if a user is searching for the hotkey for the command "Cut", he presses alt + C, and the list displays the hotkeys for the commands with the initial letter "C" (e.g. Copy, Cut, Clear and etc.). After he discovers the hotkey, he presses Ctrl + C to perform the operation of cutting.

Keyboard-based visual hotkey display techniques

5. Touch Screen Keyboard

Image 7: The keyboard mode of a touch screen keyboard

A touch screen keyboard is a keyboard displayed on a touch screen. Different from a traditional keyboard, it is not only an input device but also a display device, which enables it to display hotkeys. The keyboard has different modes and the users can switch between these modes. They access the hotkey mode by pressing the modifier key. Image 7 shows a touch screen keyboard in the keyboard mode. By pressing the modifier key "Hotkey" positioned in the lower left corner of the keyboard, the keyboard mode is switched to hotkey mode which is depicted in Image 8.

Image 8: The hotkey mode of a touch screen keyboard

The hotkey mode in Image 8 contains five menus and under each menu there are five hotkeys. When a menu is selected, the background light of the hotkeys under this menu switches on. The menus can be hierarchical to contain more hotkeys if an application

has many commands. By pressing the hotkey “BACK TO KEYBOARD”, the users switch the hotkey mode to keyboard mode.

6. Keyboard with displaying Keys

In the hotkey mode, each key on the keyboard is able to display any image or text and be assigned with a command. Thus, the keys on the keyboard change to hotkeys after a user presses a modifier key. This kind of a keyboard is already available on the market. Below is a picture of a keyboard produced by the company Optimus Popularis.

Image 9: The Optimus Popularis compact keyboard

7. Flashing Hotkeys

This technique displays hotkeys on the keyboard. After a user clicks on a command in the menu, the modifier key(s) and the letter key that form the corresponding hotkey start to flash. The user needs to press the keys to stop the flashing. For example, if a user clicks on the command “Copy” in a dropdown menu, the keys “Command” and “C” that form the command’s corresponding hotkey will start to flash, as depicted in Image 10. The user needs to press Command and C to stop the flashing. Therefore, the flashing keys serve as visual stimuli through which the hotkeys are displayed. They create visual exposure to the hotkeys and disabling flashing keys serves as a practice of using hotkeys.

Image 10: A keyboard with flashing hotkeys

An auditory channel is the other alternative modality for displaying hotkeys. An auditory display provides a bidirectional, communicative connection between the users and the hotkey display. It is based on an auditory interface which involves machine listening, speech recognition, and a dialog system [34]. The auditory interface takes the users' speech as input and displays the hotkeys through speech as output. Two auditory display techniques are developed for displaying hotkeys.

8. Voice Activated Display

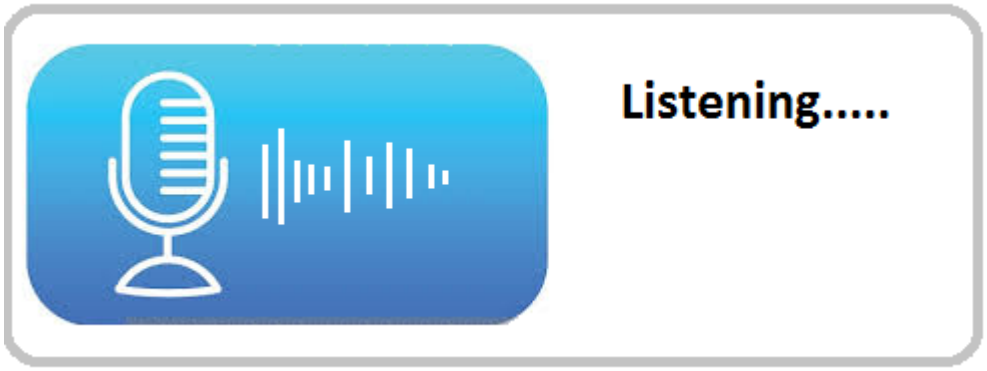

Image 11: An auditory interface for displaying hotkeys

Voice activated display provides access to hotkeys through voice recognition technologies. A user presses a modifier key to activate the auditory interface which is depicted in Image 11. Then he vocalizes the command through a microphone and the speech is recognized by the system. Finally, the corresponding hotkey is displayed vocally on voice equipment such as a headset or speakers. For example, a user says "Copy" to the auditory interface which processes the word and then generates a vocal reply of the corresponding hotkey "Copy is Command plus C".

9. Voice Command

Voice command takes the auditory display a step further. Instead of replying to the users with the corresponding hotkeys, the voice command interface directly performs the operation. For example, after a user presses down the modifier key, he activates the user interface of the voice command that is depicted in Image 11. He says “Copy” to the auditory interface, the word is recognized as a command and the program copies the content to the clipboard. Finally, the voice command interface responds with a resulting “Copied” or “Failed to copy” to indicate the outcome of the operation.

3.2 Evaluation of the Designs

The nine hotkey display techniques are grouped into different categories according to their features. First, they can be categorized into visual and auditory display according to the modality through which the hotkeys are displayed. Regarding the visual display, the techniques can be further divided into two sub-categories: keyboard-based and screen-based display. The evaluations first take place on the category level. If a category is not selected, then the sub-categories under it will not be evaluated.

3.2.1 Auditory Display vs. Visual Display

Auditory display techniques have been applied to various kinds of devices, such as alarm clocks and GPS navigation devices. This part of the chapter evaluates the advantages and limitations of auditory display to determine whether it is suitable for hotkey display.

3.2.1.1 Advantages of Auditory Display

The most important feature of the auditory display is that it does not require any use of visual resources. The users are often working on computers in multi-tasking environments, and their visual resources are allocated to the tasks displayed on the computer screens. If they need to access the hotkey display visually, they have to temporarily move their vision away from the task areas and search for the hotkeys. Such division of attention between the visual channels is defined as intra-modality [35]. If the hotkey display is positioned far from the task area, visual scanning between the visual channels introduces search cost. If they are too close to each other, this may lead to visual confusion and masking. Therefore, performing several time-shared visual tasks causes performance decrement [36]. Therefore the hotkeys can be displayed vocally to avoid visual overload.

Compared to visual interface, the auditory interface occupies much less space. Since the hotkeys are displayed vocally, no list is required, which saves precious screen space. Searching hotkeys through the auditory interface is more efficient. Computer application often contains tens of commands. Going through the hotkey list is time consuming and sometimes fruitless. The auditory interface provides a straight forward approach by enabling the users to vocally request for the hotkeys. Another advantage of

the auditory hotkey display is that it can utilize the Question Answering (QA) technology [37] which can make the auditory interface more intelligent. This technology enables the auditory interface to provide the hotkeys based on the users' descriptions of what they want to accomplish instead of the exact names of the menu items. For example, a user wants to block advertisement on the current web page, and he does not know the name of the command with such function. With the support of QA technology, auditory interface allows him to describe the function of the command (i.e. to block advertisement) and provides the corresponding hotkey.

3.2.1.2 Limitations of Auditory Display

Auditory memory has proven to be less robust than visual memory. Several experiments used delayed recall and recognition paradigms to compare auditory and visual memory, and concluded that remembering auditory stimuli is more difficult than remembering visual stimuli [38]. Therefore, the users' learning of hotkeys is less efficient if the hotkey display is auditory. Simultaneous sound waves traveling closely can cause interference. If the users are working in noisy surroundings, there is a high risk that their voices will be neutralized by the background noises, which will bring great challenges to the auditory interface's accuracy of recognizing the users' vocal inquiries.

Privacy is another concern of auditory display. Sound waves are omnidirectional which means they cannot be directed towards one particular user. The users will reveal what they are doing on the computers and the sounds may be disturbing to the people around them. This may discourage the users from practicing the use of hotkeys. Another disadvantage of auditory display is its impermanence. Unlike a visual display that stays on a computer screen until the user deactivates it, auditory display disappears after the hotkey is voiced. If the users do not have enough time to encode the hotkey, they have to repeat the process. This violates the design criteria of providing long enough exposure to hotkey display and fails to catch the users' attention.

While the auditory display has exhibited great features, its disadvantages determine that it cannot replace visual display for hotkeys. The users often work in shared spaces. In order to overcome the noises from the surroundings, they may have to bring headsets with them. Moreover, the auditory interface broadcasts what the users are doing in an undesirable manner. Furthermore, few tasks require the users to concurrently search for hotkeys while performing these tasks. Therefore, visual display is a more suitable approach for displaying hotkeys.

3.2.2 Screen-based Display vs. Keyboard-based Display

Traditional linear menus display hotkeys next to the commands and once the users click on the commands, the hotkey display disappears. Such designs hardly draw the users' attention to the hotkey display. Moreover, screen-based display of hotkeys creates visual overload to the already saturated screen space. Switching between tasks and the hotkey display leads to constantly changing scenes on the screens which can be distracting and limit the amount of information the users receive. Therefore, the hotkey display should be removed from the screen to reduce visual congestion.

Keyboards have traditionally been used as input devices. The emerging touch screen technology and LED technology enable keyboards to possess multiple functions such as displaying information visually. Compared to a screen-based display, displaying hotkeys on the keyboard eliminates the user's need to search for the hotkeys on the screen with a mouse after which he needs to move his attention to the keyboard. This is a great advantage especially when the users are working on tasks such as word processing during which their hands are positioned on the keyboard. From the perspective of innovation, the keyboard-based hotkey display is a new technique. When the users notice that their keyboards are not only input devices but are also able to display hotkeys, they will become curious about this new mechanism. This curiosity motivates them to practice using hotkeys. Therefore, a keyboard-based hotkey display is more effective than the traditional screen-based hotkey display. This leaves two keyboard-based hotkey display techniques for the final selection.

3.3 Selection between Flashing Hotkeys and Touch Screen Keyboard

Displaying hotkeys through flashing hotkeys or on a touch screen keyboard have their respective advantages and limitations. The processes through which a user interacts with the techniques are modelled below to obtain the features of these two hotkey display techniques. Finally, each technique's features are compared to the design criteria to decide which one is the optimal technique for displaying hotkeys.

3.3.1 Model Overview of Touch Screen Keyboard

The task of invoking a hotkey on a touch screen keyboard can be divided into two consecutive subtasks: visual search and finger pointing. The total selection time of a hotkey is the sum of the time spent in the visual search of the hotkey and the time spent in the motor action of finger pointing. Both subtasks are affected by learning. The learning model is a power law of practice model [39] which is parametrized for each search component. The performance time T is given:

$$T = A + B\, e^{(-\alpha N)} \qquad (1)$$

where A (> 0) is asymptote of learning as N goes infinitely, N is the trial number, B is performance time on the first trial (N = 1), and α is the learning rate. The negative exponential form indicates a decreasing curve.

The first subtask (i.e. visual search) uses eye gaze and memory recall to locate the target hotkey's location. There are two possible searching strategies [40]:

- Serial search, which refers to the top-to-bottom scanning of the hotkeys contained in a menu. Practice enables the users to skip more hotkeys and spend less time on each visited hotkey. The time spent in serial search is denoted by T_S.

- Directed search, which refers to the users' attempt to gaze directly at the target hotkey. Initially, such attempts are random but they become more accurate

through practice. Once the users are familiar with the layout of the hotkeys, they can recall the general location of the target hotkey, and land their gaze at the neighboring area of the target hotkey and eventually locate it. The time spent in directed search is denoted by T_D.

The second subtask (i.e. finger pointing) consists of the motor actions of a user pointing his finger on the target hotkey and pressing it. In HCI, Fitts's law has proven to be a robust model for predicting the completion time of human motor behavior. It is typically defined as [41]:

$$T = a + b \log_2 (1 + \frac{D}{W}) \quad (2)$$

where T is the average time spent in accomplishing the task, D is the distance from the starting point to the center of the target, W is the width of the target, a and b are constants reflecting the efficiency of the pointing system.

While Fitts's law has been serving successfully as a quantitative foundation for human-computer interaction research for decades, it is inaccurate to apply Fitts's law to finger pointing on touch screens because target acquisition tasks have typically been conducted with a stylus or a cursor. Compared to the size of a cursor or a stylus, the sizes of the targets are much bigger. However, for finger pointing on touch screen keyboards, the target hotkey's size is not much bigger or even smaller than the size of a finger. Several experiments have shown that the predictive power of Fitts's law decreases when the targets are small [42]. Therefore, Fitts's law needs to be expanded to predict finger pointing performance in small target acquisition tasks.

The subtask of selecting a hotkey is finger pointing, which refers to the time that eyes spend on the target hotkey to press it with a finger after the hotkey is located. This is predicted by FFitts's law [43] which is an expansion of Fitts's law for finger input:

$$T_P = a + b \times log_2 (1 + \frac{D}{\sqrt{2\pi e(\sigma^2 - \sigma_f^2)}}) \quad (3)$$

where σ is the standard deviation of the distribution of endpoints which can be measured directly, and σ_f reflects the absolute precision of the input finger which may vary according to the individual's finger size.

The total selection time T is the sum of serial search time T_S, directed search time T_D, and finger pointing time T_P and is given as:

$$T(\theta) = T_S(\theta) + T_D(\theta) + T_P(\theta) \quad (4)$$

where $\theta = (l, t, P)$ is a vector containing four input variables for the model: l denotes the length of the menu in which the target hotkey is located, t denotes the target hotkey's position, and P denotes a vector containing the number of how many times the target hotkey has been visited previously [40].

There are six factors about the hotkey display on a touch screen that affect the target hotkey's selection time [40]:

1. The length of the menu: the target hotkey's selection time is proportional to the amount of hotkeys grouped under the menu. The users have to spend more time searching for the target hotkey in a longer menu.
2. Hotkey organization: hotkeys can be grouped under menus according to certain rules. The example in Image 8 is organized according to the functions of the hotkeys. Searching for the target hotkey from unordered hotkeys will take more time.
3. The target hotkey's position: if the target hotkey is positioned close to the top, it is faster to select it, assuming the users start from the top and then scan the hotkeys one by one.
4. Absent target hotkey: going through a menu that does not contain the target hotkeys takes more time than going through one that contains it.
5. Practice: if the layout of the hotkeys is stable, the users can remember the approximate locations of the hotkeys and carry out a directed search. Therefore, selection time reduces when the users practice more.
6. Finger movement: the users mainly select between two strategies to coordinate the finger movement with visual guidance: i) once the target hotkey is located, the user makes a single move to point his finger onto the target hotkey and ii) the user's finger follows the eye gaze and hovers over the hotkeys [40].

3.3.2 Model Overview of Flashing Hotkeys

This model is used to derive the learning process of hotkeys thorough the flashing hotkey display technique. Each time a user presses a hotkey either to accomplish the task or to stop the flashing, a memory trace is created and utilized in the next task to reduce the retrieval time of the hotkey from the user's Long Term Memory (LTM). The model assumes that if the hotkey's retrieval time is shorter than the completion time of pointer-based menu selection, its expected utility is larger than the menu selection's expected utility. Therefore, a user will adopt hotkey use. In this model, the command's corresponding hotkey is assigned randomly. Therefore, the impact of intuitive hotkey assignments [15] is not considered.

The model is comprised of four components which are depicted in Figure 5. It portrays a user's decision-making process concerning the selection between the two competing strategies: pointer-based menu selection and hotkey.

1. Menu selection: The motor operations required to accomplish a task through pointer-based selection of a command is configured by Keystroke Level Model (KLM) [41]. The task completion time of pointer-based menu selection is the sum of the time spent in these motor operations.

2. Hotkey retrieval: Once a user disables a flashing hotkey for the first time, an entry associating the command and its corresponding hotkey has been created. The entry has a base level activation Bi. Every time the user practices using the hotkey, the command-hotkey association's base level activation increases. The equation for the base level activation is given:

$$B_i = ln\left(\sum_{j=1}^{n} t_j^{-d}\right) \quad (5)$$

where t_j is the time since the j:th visit of i, d is the decay parameter. This equation reflects the log odds of revisiting a hotkey as a function of how many times it has been visited previously. It produces both the power law of forgetting and the power law of learning. The power law of forgetting is reflected by the negative exponential form (-d) that produces a decay effect, where the strengths of the previous visits of the hotkey fade away as power functions. The power law of learning is reflected by the summation of the strengths of the previous visits of the hotkey, where individual visits accumulate strength as a power function of the number of visits [44].

The activation of the entry presents a preliminary estimate of how available the information will be. It is the sum of its base level activation and associative activations [44]. The activation equation is given as:

$$A_i = B_i + \sum_j W_j S_{ji} \quad (6)$$

where B_i is the base level activation of element i which reflects the usefulness of i in the past. W_j is the source activation of element j in the focus of attention, and S_{ji} is the strength of association of from element j to element i.

The activation of the entry controls the speed of retrieving the hotkey. The latency of retrieval equation is given as [44]:

$$T_i = Fe^{-fAi} \quad (7)$$

where F is the latency factor, and f is a scaling constant. This equation maps activation onto retrieval latency.

The model predicts high retrieval latency in the beginning when the users have little experience with the hotkeys. By practicing through disabling the flashing hotkey, the hotkey's retrieval latency is reduced. Eventually, the hotkey's expected utility exceeds the one of the pointer-based menu selection.

3. Selection: This component determines which of these two competing strategies will be selected. The decision is based on expected utility calculation. If a production rule is executed at time (n – t) and it will receive a reward at time n, then the reward term in utility learning, R(n), is computed as [45]:

$$R(n) = r - t \quad (8)$$

where r is the reward, t is the task completion time.

After a task is accomplished, the utility is updated via the difference learning rule [45]:

$$U_i(n) = U_i(n - 1) + \alpha[R_i(n) - U_i(n - 1)] + \sigma_u \quad (9)$$

where α is the learning rate, U_i (n - 1) is the utility of the previous task. σ_u is the noise or uncertainty for choosing between pointer-based menu selection and the hotkey. The noise is from a normal distribution with standard deviation σ.

4. Execution: If the hotkey is selected to accomplish the task, the motor actions of pressing the hotkey will be executed. If the task is accomplished through the pointer-based menu selection, users will execute the pressing of the hotkey to stop the flashing.

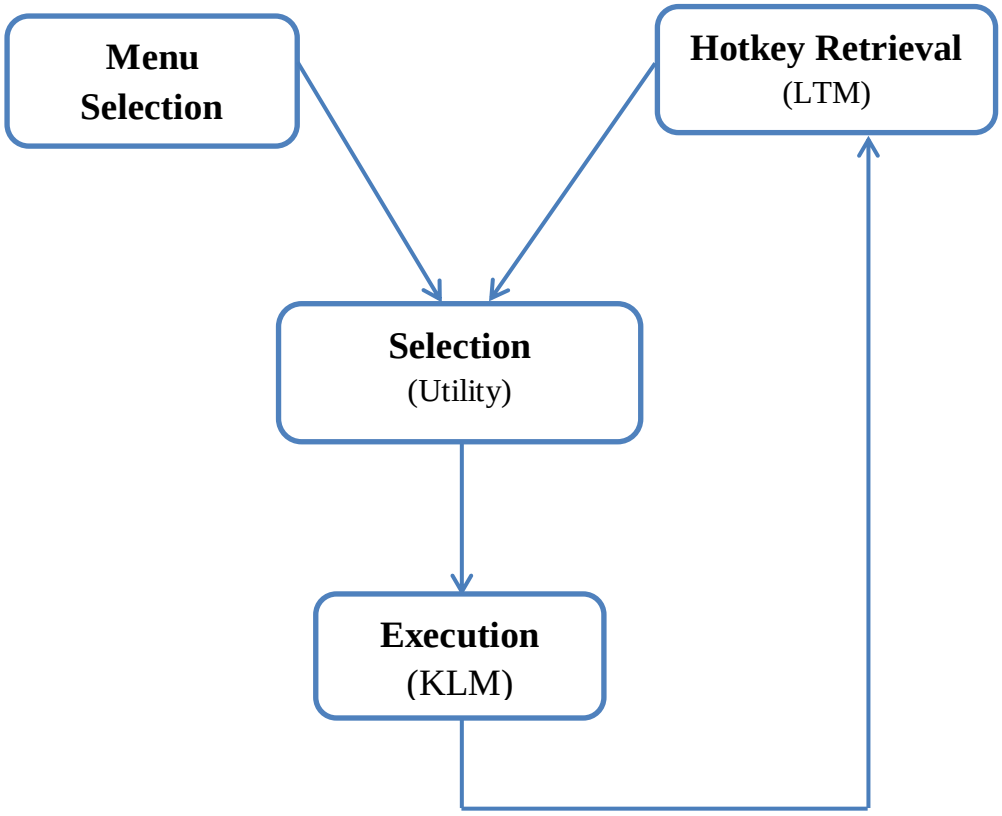

Figure 6: The model of learning hotkeys through flashing hotkeys.

Figure 6 shows how a user learns the hotkey through the flashing hotkey display technique. Initially, the user is more likely to choose pointer-based menu selection to accomplish the task since it has higher expected utility. After he clicks on the command, he needs to press the hotkey to stop it from flashing. Subsequently, the user practices using the hotkey regardless which strategy he employs. Practice increases the hotkey's activation. Eventually, the hotkey's retrieval latency is shorter than the completion time of menu selection and the user will adapt hotkey use to obtain higher expected utility.

3.3.3 Flashing Hotkeys vs. Touch Screen Keyboard

The model of invoking a hotkey from the hotkey display on a touch screen keyboard shows that the total selection time of a hotkey is the sum of the serial search time, the

directed search time and the time spent in the motor action of finger pointing. Inexperienced users can locate the target hotkeys through serial search. After they have gained sufficient amount of experience, they remember the general position of the target hotkey so that they can glance at the adjacent area of the target hotkey to quickly locate it. Therefore, invoking the target hotkey is not dependent on the successful memory recall of the hotkey's location, which greatly reduces the memory demand for using hotkeys and is favored by the minimum memory hypothesis [46].

The model indicates that accomplishing a task through a hotkey on a touch screen is comprised of two subtasks: the visual search of the target hotkey and the motor action of pointing. Therefore, this strategy is based on "see and point" [7] and users are benefited from the direct manipulation principle which allows them to visually locate the target hotkeys and directly manipulate them through pointing with fingers. This approach is also used in linear and toolbar menus. While displaying hotkeys on a touch screen keyboard provides the benefits of "see and point", it also manifests the limitation accompanied with the direct manipulation principle. After a user becomes experienced with the layout of the hotkeys displayed on a touch screen, he can locate the target hotkey through directed visual search and shorten the task completion time. However, the improvement of the user performance brought by directed search is limited. Like linear and toolbar menus, invoking hotkeys displayed on a touch screen keyboard cannot support expert performance, which means that users who pursue expert performance will not adapt this technique. Additionally, this technique does not support incidental learning. The system needs to provide an instruction to make users aware of the existence of the new hotkey mode. Users press the modifier key "Hotkey" to switch the keyboard to hotkey mode. The saliency of this modifier key is not higher than the other keys. Therefore, it is uncertain that this technique will attract novice users.

Displaying hotkeys through flashing keys supports expert performance. The equation of base level activation reflects the impact of practice. This equation shows that the base level activation of a hotkey increases every time after a user invokes it. The increment of the base level activation leads to the increment of the activation, which in turn, reduces the retrieval latency of the hotkey. Therefore, the more a user invokes the hotkeys, the faster he retrieves them from his LTM. Hotkey display through flashing keys creates sufficient exposure through the visual stimuli that display hotkeys and catch the users' attention. This technique displays hotkeys without relying on users' intentional access to the hotkey display and supports incidental learning. After a user invokes a command through pointer-based menu selection, he is reminded of the command's corresponding hotkey through the flashing of the keys. This mechanism provides users with the opportunity to learn and relearn hotkeys. However, all these advantages come with a cost of heavy memory load.

Selecting between these two hotkey display techniques means a tradeoff between the strategy that minimizes the memory demand by repeatedly interacting with the task environment and the strategy that minimizes interactions by increasing the memory demand. In fact, expert performance is not pursued by all of the users. For users who prefer minimum demand on memory, hotkey display on a touch screen keyboard is more suitable. On the other hand, supporting expert performance is a main feature of

hotkeys. For users who require efficiency, hotkey display through flashing keys is optimal.

To help select between these two strategies, a table is constructed to compare them to the design criteria. As shown in Table 2, displaying hotkeys through flashing keys fulfills more design criteria and thus, is selected for further study in this thesis. The next chapter presents a detailed model of how a user chooses between hotkeys and pointer-based menu selections when the flashing hotkey display technique is used.

Design Criteria	**Flashing Hotkey Technique**	**Touch Screen Keyboard-Based Display Technique**
Make users aware of the existence of hotkeys	YES	YES
encourage users to practice hotkey use	YES	NO
allow users to easily relearn a hotkey	YES	YES
provide a seamless switch between pointer-based selections and hotkeys	YES	YES
provide a long enough exposure	YES	YES
catch users' attention	YES	NO
avoid the use of any non-hotkey-based modality to access the hotkey display	YES	YES
supports incidental learning	YES	NO

Table 2: Comparisons of the flashing hotkey display technique and hotkey display on a touch screen keyboard.

4 Model of Hotkey Display through Flashing Keys

The human cognitive system possesses an impressive ability of managing and executing multiple tasks concurrently. Such ability is often referred to as multitasking [47]. Depending on the features of the tasks' domains, multitasking can be almost effortless (e.g. walking and talking) or exceedingly difficult (e.g. driving and checking a map). A theory called *threaded cognition* has been developed to provide a theoretical and computational framework to model and analyze the performances of concurrent execution of different tasks [47]. Compared to its alternative approaches (see e.g. [48], [49]) which fail to provide sufficient detail of instantiation across task domains, the threaded cognition theory contains a general, domain-independent model of multitasking which adequately explains multitasking between two arbitrary tasks [47] such as hotkey retrieval and pointer-based menu selection.

When a user is choosing between a pointer-based menu selection and a hotkey to accomplish a task, there are two different decision-making processes:

1. Concurrent dual-tasking, in which a user tries to retrieve the hotkey from his Long Term Memory while executing the motor operations of pointer-based menu selection. If the user successfully retrieves the hotkey before he completes the motor operations, he will select the hotkey to accomplish the task.
2. Sequential tasking, in which a user first tries to retrieve the hotkey for a short interval of time. If he fails to retrieve the hotkey within the time interval, he will use pointer-based menu selection to complete the task. In this chapter, the threaded cognition theory is applied to model a user's decision-making process of concurrent dual-tasking. The pointer-based menu selection and hotkey retrieval are put within the framework of threaded cognition to demonstrate that they can be simultaneously executed without interference. Thus, menu selection and hotkey retrieval are presented as two threads of processing which are coordinated by perceptual, motor and declarative recourses. The subtask that requires less completion time can maximize the expected utility [50] and thus, is rationally selected by the user to accomplish the task.

This chapter also contemplates on modelling a hotkey's retrieval latency when a user is attempting to retrieve multiple hotkeys in a row. The model will consider the effects of users' working memory limitations and reassignments of modifier keys and letters. These effects have a direct impact on a hotkey's retrieval latency. Finally, the predictions of hotkey retrieval latencies are used to simulate the users' decision-making processes of choosing between hotkeys and menu selection in the form of concurrent dual-tasking.

4.1 Concurrent Dual-tasking of Menu Selection and Hotkey Retrieval

The threaded cognition theory states that cognition can maintain and execute multiple active goals, resulting in concurrent threads of resource processing [47]. It focuses on deliberative multitasking in which concurrent tasks are performed on a second time scale, which ensures that it is applicable to meu selection and hotkey retrieval. According to threaded cognition, pointer-based menu selection and hotkey retrieval are two threads which have their respective goals that are maintained in the active goal set. The goal for menu selection is to move the cursor onto the target command and click on it. The goal for hotkey retrieval is to retrieve the target hotkey from Long Term Memory (LTM).

Each goal has relevant production rules (often referred to as rules). A rule defines a set of conditions and actions in such a way that the conditions must be fulfilled for the rule to execute the actions. The menu selection thread and the hotkey retrieval thread do not require the same peripheral resource (e.g. vision). Figure 7 shows the required resources for pointer-based menu selection and hotkey retrieval.

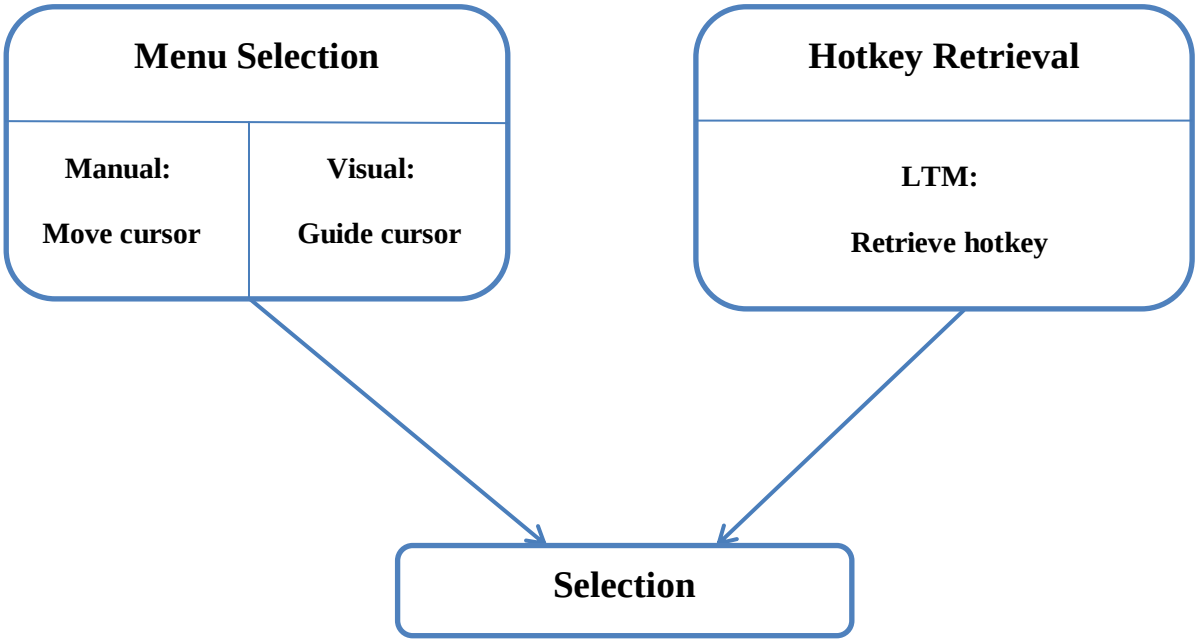

Figure 7: The threads and resources of menu selection and hotkey retrieval

As depicted in Figure 7, when these two threads are executed in parallel, the hotkey retrieval thread requires declarative resource. The menu selection thread incorporates perceptual resource which visually guides the cursor and motor resource used to manually move the cursor.

Moreover, these two threads do not require attention from the procedural resource (e.g. when visual and auditory stimuli are encoded simultaneously and require the procedural resource to redirect the information to proceed). They are autonomous process threads

which do not interact with each other. Therefore, the threads do not raise contention for each other's resources and can be executed comfortably in parallel without any dual-task interference or any contention which restrains them [47].

4.2 Selection between Two Competing Threads

The menu selection thread and the hotkey retrieval thread are processed concurrently. The one that first reaches the selector requires less processing time and maximizes the expected utility. Therefore, it will be selected to accomplish the task. In other words, a user moves the cursor onto the menu, clicks on it and then moves the cursor onto the command and clicks on it. The total completion time of these motor operations is modelled with KLM. At the same time, the user attempts to retrieve the command's corresponding hotkey from LTM, which is calculated by a retrieval latency equation. If the user manages to retrieve the hotkey before he clicks on the command (i.e. the hotkey retrieval thread reaches the selector first), the menu selection thread is terminated and the task will be accomplished through the hotkey. The selection between the hotkey and menu selection can be concluded by the equation:

$$\textit{Thread processing time} = \min (T_{menu}\ ,\ T_{retrieval}) \qquad (10)$$

where T_{menu} is the thread processing time of pointer-based menu selection, and $T_{retrieval}$ is the thread processing time of hotkey retrieval. The thread which requires less processing time is selected and the other one is terminated. This model uses utility learning to predict a user's choice between these two threads to accomplish a given task. The utility of a strategy after its *n*th application is calculated through the difference learning equation [45]:

$$U_i(n) = U_i(n - 1) + \alpha[R_i(n) - U_i(n - 1)] + \delta_U \qquad (11)$$

where $U_i(n - 1)$ is the utility of a strategy after its n – 1st application, and $R_i(n)$ is the reward the strategy receives after its *n*th application, which is the external reward received minus the execution time of the strategy. Therefore, the thread which takes less time maximizes the utility and will be selected. A noise is added into this equation to account for the possibility that the thread processing times are so close that the user randomly selects between these two threads. The noise is from a normal distribution with a standard deviation δ_U.

4.3 Math Models for Pointer-based Menu Selections and Hotkey Retrievals

In this section, Mathematical models are constructed to predict execution times of menu selection, hotkey pressing and hotkey retrieval. These models are used to simulate two learning scenarios: learning one hotkey and learning multiple (5) hotkeys. Learning hotkeys is defined as the process through which relatively permanent transitions from

menu selections to hotkeys occur as a result of past experience [11]. In other words, a user who is accompanied to using commands for tasks practices hotkeys by disabling the flashing modifier key(s) and the letter key. The user has learned the hotkeys when the retrieval latencies are shorter than the menu selection times and he transits from menu selections to hotkeys.

The motor processes of menu selection and pressing a hotkey are modelled after Keystroke Level Model (KLM) which provides an accurate estimation of the completion times for the tasks. The retrieval latency of a hotkey is based on Anderson's list memory model [26] which addresses latency patterns and the probability of successful retrievals. Finally, the effect of working memory limitations is modelled to simulate the transition processes from menu selection to hotkeys of users with different working memory capacities.

4.3.1 KLM for Pointer-based Menu Selection

The menu selection's execution time is predicted by KLM. The users must perform a sequence of keystroke-level actions to accomplish menu selection. The menu selection time is the sum of the times spent on the actions.

4.3.1.1 A Menu with One Item

Selecting the command from a menu containing only one item is the simplest scenario. The operator sequence of menu selection and each operator's execution time [51] is listed below:

1. Home hand to the mouse (T_H = 0.4 s)
2. Point mouse to the dropdown menu (T_P = 1.76 s)
3. Press mouse button (T_B = 0.1 s)
4. Release mouse button (T_B = 0.1 s)
5. Point mouse to the command (T_P = 0.19 s)
6. Press mouse button (T_B = 0.1 s)
7. Release mouse button (T_B = 0.1 s)

Operator 2 and 5 are predicted by Fitts's law [51] for more accurate estimations:

$$T = a + b \log_2 (1 + A/W) \quad (12)$$

where a and b are constants that depend on the choice of input device and are usually determined empirically by regression analysis. For pointing with a mouse, *a* is set to 108 ms and *b* is set to 392 ms [51]. A is the distance from the starting point to the center of the target. W is the width of the menu measured along the axis of motion. For

operator 2, A is assumed to be 15 cm (from the center of a screen to the menu's center) and W is assumed to be 0.85 cm. For operator 5, A is 0.6 cm (from the menu's center to the item's center). W is assumed to be 3.8 cm. The widths are measured on Firefox's Edit menu. The total time for menu selection is given:

$$T_{menu} = T_H + T_P + T_B + T_B + T_P + T_B + T_B = 2.75\ s$$

This result is based on the physical operators of menu selection. Other possible operators such as thinking and perception are not considered.

4.3.1.2 A Menu with Multiple (5) Items

Selecting an item from a menu containing multiple items involves an extra operator: mental act of routine thinking or perception, M, because when users perform menu selections, there are pauses in the stream of actions which are associated with visually searching for the commands. Based on the available research [52], a reasonable overall estimation for the duration of visual search is 0.8 second.

Operator sequence of menu selection is listed below:

1. Home hand to the mouse (T_H)
2. Point mouse to the dropdown menu (T_P)
3. Press mouse button (T_B)
4. Release mouse button (T_B)
5. Visually search for the command (T_M)
6. Point mouse to the command (T_P)
7. Press mouse button (T_B)
8. Release mouse button (T_B)

Since there are five items in the menu, the execution time of operator 6 is determined by the target command's position in the linear menu.

The following is an example of a menu with five items, their positions, their corresponding hotkeys and the selection time for each item:

Command	Position	Corresponding Hotkey	Selection Time
Copy	1	Ctrl + A	3.55 s
Cut	2	Ctrl + B	3.62 s
Close	3	Ctrl + C	3.69 s
Print	4	Ctrl + ALT + C	3.74 s
Edit	5	Ctrl + ALT + D	3.80 s

Table 3: A linear menu with five items and their respective selection times

The commands and their corresponding hotkeys listed in Table 3 are used to model the retrieval latencies and simulate the learning processes of one hotkey and multiple hotkeys.

4.3.2 KLM for Hotkey Pressing

KLM is also used to predict the motor process of pressing the hotkey. In this thesis, a hotkey is a combination of a modifier key or keys and a letter key. By pressing the hotkey, the users can complete a task or disable the flashing keys. The key stroke time is estimated on the experience level of average non-secretarial typists which is 0.28 seconds for pressing one key. The operator sequence for a hotkey is:

1. Home hand to the keyboard (T_H = 0.4 s)
2. Press the modifier key (T_K = 0.28)
3. Press the letter key (T_K = 0.28)

The total time for a hotkey stroke is given as:

$$T_{HK} = T_H + T_K + T_K = 0.96\ s$$

where T_{HK} is the total time spent in pressing the hotkey.

4.3.3 Models of Hotkey Retrievals

The ACT-R theory is a theory which models human cognition and assumes that a production system operates on a declarative memory [53]. It emphasizes that the production system can be related to the declarative memory through an activation-based processing mechanism. The activation level determines the speed and probability of a trace in declarative memory being processed by the production rule [53].

According to ACT-R, declarative knowledge, such as a command's corresponding hotkey, is represented by a schema-like structure called chunk [53]. Figure 8 illustrates the knowledge chunk of the fact that the corresponding hotkey for the command "Copy" is "Ctrl + C":

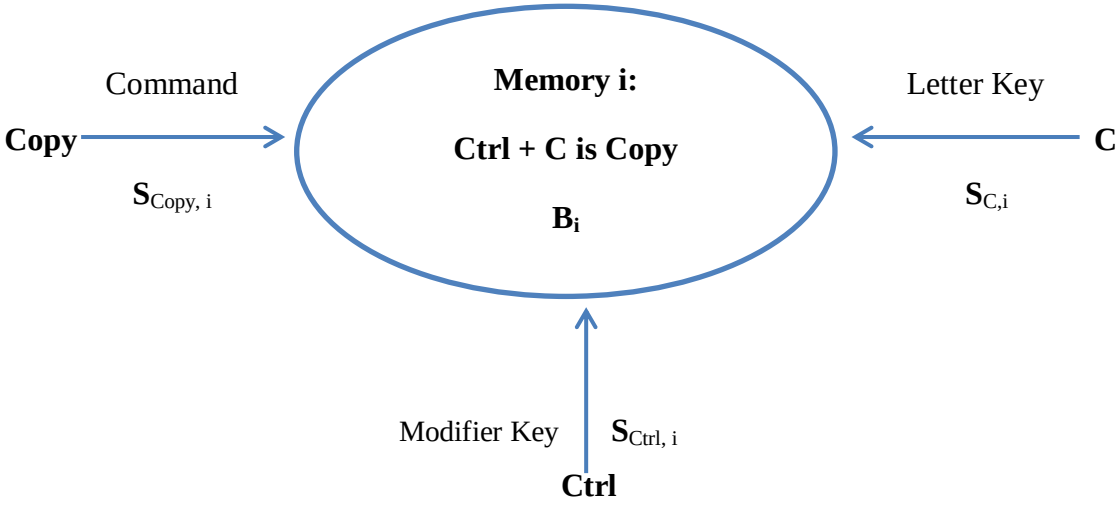

Figure 8: A graphic network representation of the chunk encoding a *fact* that "Copy is Ctrl + C". The oval represents the *fact* which encodes the command "Copy" and its corresponding hotkey. The elements outside of the oval, "Ctrl", "C", and "Copy", represent the *concepts* which are potential sources of activation. The oval has some relatively stable base-level activation B_i and also receives activation from the elements connected to it according to the strength of association S_{ji}.

The speed and probability of retrieving such a chunk is determined by its total activation level which is calculated as the sum of the chunk's base-level activation B_i and the amount of source activation it receives from elements currently in the focus of attention. The equation is given as:

$$A_i = B_i + \sum_j W_j S_{ji} \quad (13)$$

where B_i is the base level activation of memory i which reflects the number of practice of i in the past. W_j is the source activation of element j in the focus of attention, and S_{ji} is the strength of association of from element j to element i [53].

The summation, $\sum_j W_j S_{ji}$, is over the concepts, j, of the probe which provides sources of activation [54]. W_j is the amount of available source activation which is limited [55]. The limitation is given as:

$$\sum W_j = 1 \quad (14)$$

This equation shows the limitation on the amount of attention a user can distribute over the source objects, which explains why the user's working memory is limited [55]. The values of W_j vary according to each individual and the distribution of these values follows a normal distribution which is centered at 1 [27]. In this way, how each individual's limited source activation affects his learning of hotkeys can be simulated by assigning different values to W_j.

As shown in Figure 8, a user's source activation is divided equally among the three elements in the current focus of attention. It spreads from the elements to the chunk which is mandatory for performing the task "Copy" with the hotkey. In this way, task-relevant knowledge chunks are maintained in an available state relative to the rest of the declarative memory [55]. Therefore, equation 13 can also be given as:

$$A_i = B_i + \sum \frac{W}{n} S_{ji} \tag{15}$$

where W is the amount of available source activation, and n is the number of elements currently in focus of attention. Regarding the command "Copy" and its corresponding hotkey, there are three sources of activation: the modifier key "Ctrl", the letter key "C" and the command "Copy", which is depicted in Figure 8.

The other factor in the summation, S_{ij}, is the strength of association between the fact *i* and each of its associated concepts. For example, as regards the hotkey for the command "Copy", S_{ji} includes $S_{Copy,\ i}$, $S_{C,i}$, and $\mathbf{S}_{Ctrl,\ i}$. According to Anderson and Reder [54], the facts which associate with a concept are equally likely when the particular concept is present. Therefore, the equation for strength of association can be given as:

$$S_{ji} = S - ln\ (f_j) \tag{16}$$

where S is a constant and f_j is the fan which refers to the number of facts associated with the concept *j* [54].

4.3.3.1 Model for Retrieving One Hotkey

Figure 8 shows the chunk which encodes the fact that "Ctrl + C" is the hotkey for the command "Copy". The latency for retrieving this chunk depends on its activation level A_i. In this simple scenario, the three concepts associate with the same fact, which signifies that f_j is equal to 1. Therefore, the equation for strength of association can be simplified as:

$$S_{ji} = S$$

Hence, Equation 15 is finally given as:

$$A_i = B_i + WS$$

where W is the source activation which varies according to each individual and S is the strength of association. Their product is a constant.

By combining the source activation equation and the base-level equation, the latency of retrieving a single hotkey is given as:

$$T_i = Fe^{-fWS} e^{-fln(\sum_{j=1}^{n} t_j^{-d})} \tag{17}$$

where F is the scaling factor for overall retrieval times, f is the factor for the effect of activation on retrieval time, t_j is the time since the j:th practice of the hotkey, and d is the decay parameter.

4.3.3.2 Model for Retrieving Multiple Hotkeys

Computer programs often have tens of hotkeys, which makes reassigning the modifier keys and letter keys to different hotkeys inevitable. For example, in the Safari browser, the hotkey for the command "New window" is comprised of the modifier key "Ctrl" and the letter key "N". The hotkey for the command "New incognito window" is comprised of the modifier keys "Ctrl" and "ALT" and the letter key "N". Thus, "Ctrl" and "N" are associated with two hotkeys. Therefore, one concept (i.e. the modifier keys and letter keys) might be associated with multiple facts, which complicates the calculation of each chunk's activation level A_i. The commands and their corresponding hotkeys listed in Table 3 are used to illustrate the retrieval of multiple hotkeys.

Figure 9 depicts the network between the concepts and the facts and helps to visualize how the concepts are associated with the facts. It also helps to calculate each concept's strength of association. An arrow indicates the association between a concept and a fact. Thus, the number of facts associated with a concept (f_j) is the number of arrows leading away from the concept. After f_j is figured, Equation 16 is substituted into Equation 15 to derive the activation equation which is given as:

$$A_i = B_i + \sum \frac{W}{n} (S - ln(f_j)) \tag{18}$$

Then Equation 18 is substituted into the retrieval latency equation:

$$\begin{aligned} T_i &= Fe^{-f(B_i + \sum_{j=1}^{n} \frac{W}{n}(S - ln(f_j)))} \\ &= Fe^{-f(ln(\sum_{j=1}^{n} t_j^{-d}) + \sum_{j=1}^{n} \frac{W}{n} S - \sum_{j=1}^{n} \frac{W}{n} ln(f_j))} \\ &= Fe^{-f(ln(\sum_{j=1}^{n} t_j^{-d}) + \sum_{j=1}^{n} \frac{W}{n} S)} e^{f \sum_{j=1}^{n} \frac{W}{n} ln(f_j)} \\ &= Fe^{-f(ln(\sum_{j=1}^{n} t_j^{-d}) + \sum_{j=1}^{n} \frac{W}{n} S)} \prod_{j=1}^{n} f_j^{\frac{fw}{n}} \\ &= Fe^{-f ln(\sum_{j=1}^{n} t_j^{-d})} e^{-f \sum_{j=1}^{n} \frac{W}{n} S} \prod_{j=1}^{n} f_j^{\frac{fw}{n}} \end{aligned} \tag{19}$$

Equation 19 is the equation for the retrieval latency of a hotkey when a user attempts to retrieve multiple hotkeys in a row. This equation integrates the effects of learning (B_i), working memory limitations ($\sum W$), and fans (f_j).

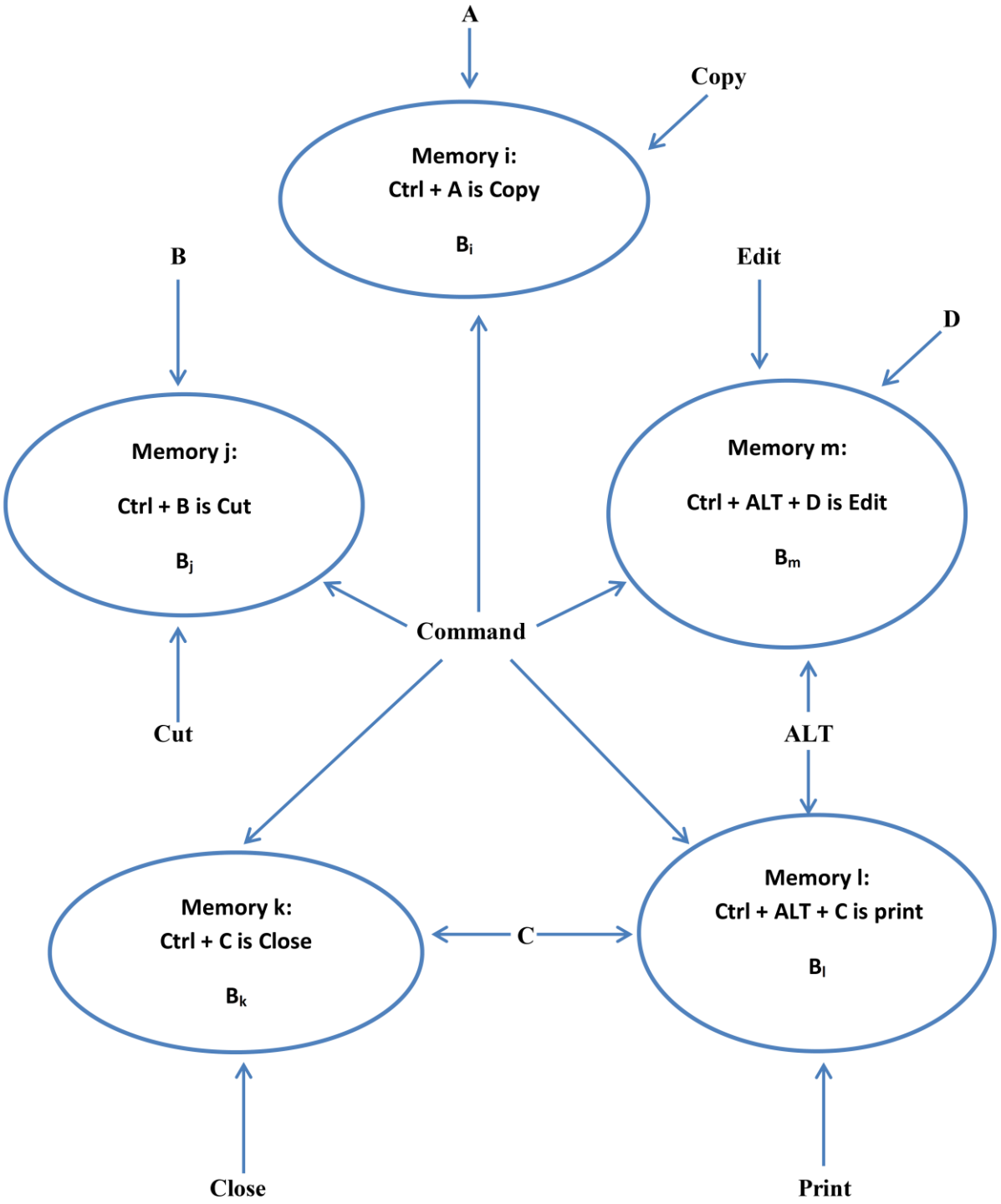

Figure 9: A network representation for the five hotkeys. The ovals represent the facts which encode the commands' corresponding hotkeys. The elements outside the ovals represent concepts associated with the facts. The arrows indicate associations between the concepts and the facts.

4.4 Simulation of Transition from Menu Selections to Hotkeys

This section simulates the processes of how users with different working memory limits learn one hotkey and five hotkeys with the help of the hotkey interaction technique which displays hotkeys through flashing keys on the keyboard. Users practice hotkeys by pressing the flashing keys to stop them. Once a user practices a hotkey, the command-hotkey paired association is encoded in the knowledge chunk. When the next task requires invoking the same command, the user chooses between the hotkey and the pointer-based menu selection which leads to flashing keys. According to threaded cognition theory, the hotkey retrieval and the motor operations of menu selection are two threads which are concurrently executed by the user. The thread which requires less execution time can maximize the expected utility and is selected to accomplish the task. Thus, the simulation is based on this assertion.

4.4.1 Simulation of Learning One Hotkey

This part aims to simulate the processes of how users with different working memory limits learn one hotkey. A program is developed in Python to simulate the concurrent dual-tasking of menu selection and hotkey retrieval as well as the increase of activation level after disabling the visual stimulus of flashing hotkeys. Finally, when the hotkey retrieval time is less than the menu selection time, users will favor the hotkeys to menu selection. The model is parametrized according to different working memory limitations. By assigning the source activation value according to a user's cognitive level, the program facilitates the study of whether users with low cognitive levels can learn the hotkey. Parameters for the model, their functionalities and their values are listed below:

Parameter	Component	Functionality	Value	Ref.
d	LTM	decay (forgetting)	0.5	[26]
F	LTM	scaling factor for overall retrieval time	1.06	[26]
f	LTM	scaling the effect of activation on retrieval time	1.53	[26]
W	LTM	source of activation	1, 0.8, 0.6	[56]
S	LTM	strength of association	2	[27]
α	Utility	learning rate	0.2	[45]
δ_U	Utility	noise for choosing between a pointer-based command and a hotkey	0.25	[45]

Table 4: Parameters for the model and their descriptions.

This program simulates the learning processes of three users whose source activation (W) values are 1, 0.8 and 0.6 respectively. 1.0 (default value for ACT-R models) is for a healthy subject, 0.8 is for subjects with poor memory and 0.6 is for subjects with impairment. These values of W are used to model the performance in Activity of Daily Living (ADL) of users on different stage of Clinical Dementia Rating (CDR) in the original study [56]. The users perform a single task of "Copy" through menu selection or hotkey (i.e. Ctrl + A) for 50 rounds. The time interval between each round is 300 seconds.

Results

The results of the simulation demonstrate that when a user attempts to retrieve a single hotkey, the user's source activation (W) and the amount of practices (Bi) have clear impacts to the hotkey's retrieval latency. The results also show that the three subjects are able to learn the hotkey. The subject with a source activation value of 1 learns the hotkey after two rounds since the hotkey's retrieval latency is shorter than the menu selection time in the beginning of the third round and the user will adopt the hotkey. For the same reason, the subject with a source activation value of 0.8 learns the hotkey after three rounds. The subject with a source activation value of 0.6 learns the hotkey after five rounds.

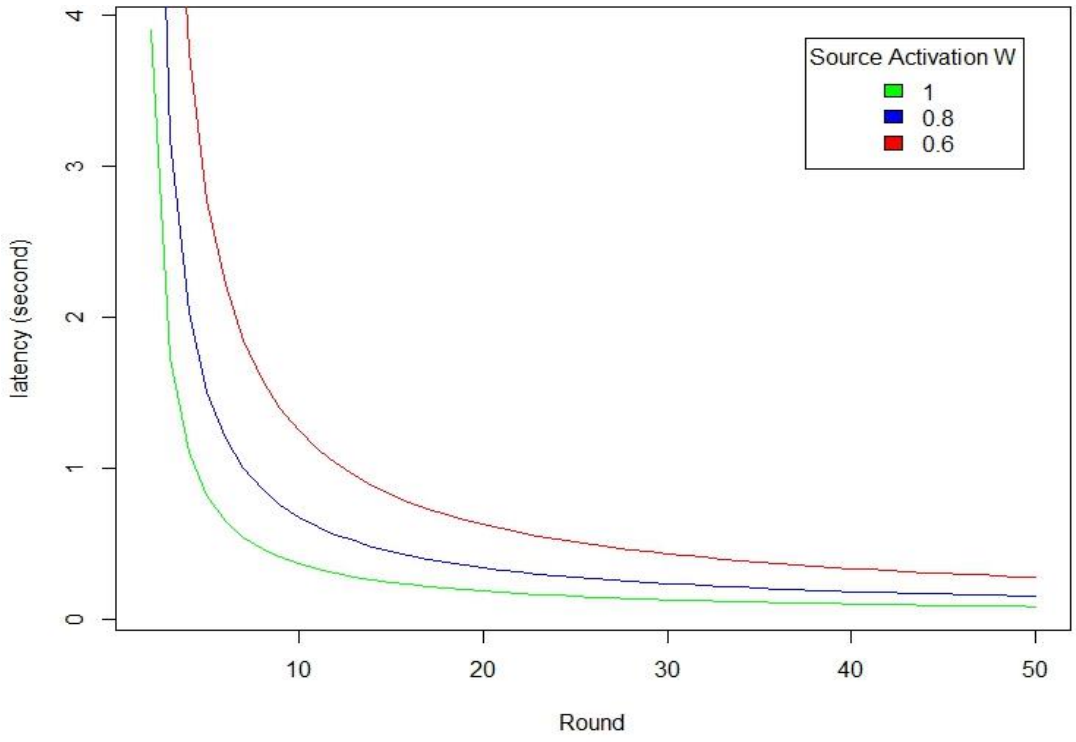

Figure 10: The Hotkey Retrieval Latencies.

As indicated by the learning curves in Figure 10, the retrieval latency increases when the value of W decreases. Subjects with low source activation values need more practice in order to learn the hotkey. The flashing modifier key and letter key serve as a visual

stimulus to help the subjects to learn hotkeys. Even the subject with memory impairment (W = 0.6) is able to learn the hotkey after practicing five rounds disabling the flashing keys.

The other factor that affects the hotkey's retrieval latency is the base level activation (B_i) which depicts a user's learning process of the hotkey. In the beginning, the three subjects have little experience with the hotkey. Therefore, the base level activation is low and the hotkey's retrieval latencies are high. The subjects practice using the hotkey through disabling the flashing modifier key and the letter key, and the base level activation increases exponentially as depicted in Figure 11.

The increase of the base level activation leads to the decrease of the hotkey's retrieval latency. In the second round, the retrieval latencies for the three subjects are 3.902s (W = 1), 7.196s (W = 0.8) and 13.272s (W = 0.6). After 50 rounds of practicing hotkey, the retrieval latencies for the three subjects are 0.080s (W = 1), 0.149s (W = 0.8) and 0.275s (W = 0.6).

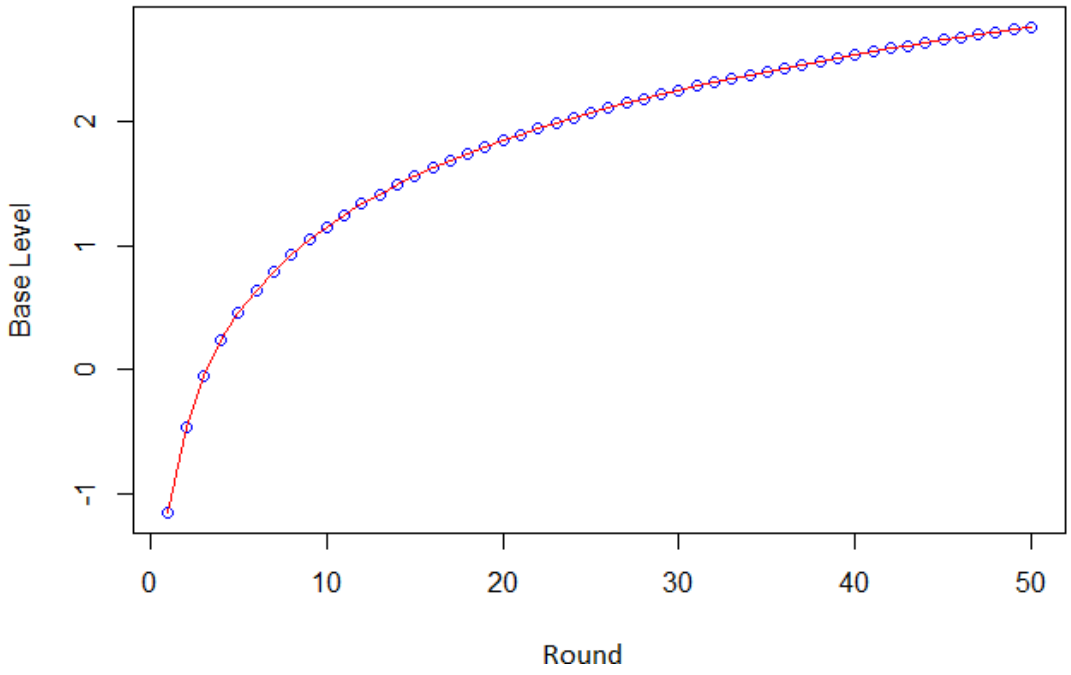

Figure 11: Base levels from round 2 to round 50.

4.4.2 Simulation of Learning Five Hotkeys

This section simulates how three subjects with different source activation values (1, 0.8 and 0.6) learn the five hotkeys listed in Table 3. A program is created in Python to simulate the learning processes of the hotkeys. Compared to the retrieval of a single hotkey, retrieving multiple hotkeys in a row is complicated by the fan effect [54] that the retrieval latency of a hotkey is affected by the number of facts associated with the hotkey's corresponding command, modifier key(s) and letter key. For instance, according to Table 3, the hotkey for the command "Print" is comprised of two modifier

keys “Ctrl” and “ALT” and a letter key “C”. “Ctrl” is associated with 5 facts, “ALT” is associated with 2 facts, “C” is associated with 2 facts, and the command “Print” is associated with 1 fact. The fan effect is modelled in Equation 18 and its impact to the retrieval latencies is simulated in the program.

The parameters used in the simulation are listed in Table 4. The program simulates three subjects’ learning processes of the five hotkeys listed in Table 3. The subjects are assigned with different source activation limitations (W = 1, 0.8, 0.6), which facilitates the study of whether users with low cognitive levelas are able to learn the five hotkeys. The users perform five tasks (i.e. copy, cut, close, edit and print) in a row and then have an intermission of 300s. Each task can be accomplished by either selecting the task-specific command or by pressing the item’s corresponding hotkey. A user learns the hotkeys through disabling them from flashing.

Facts and their associated concepts derived from the commands and their corresponding hotkeys are listed in Table 5.

Fact	**Associated Concepts / f_j**	**Number of Elements in Focus (n)**	**Source Activation**
Ctrl + A is Copy	Ctrl / 5 A / 1 Copy / 1	3	1.46 （W = 1） 1.168（W = 0 .8） 0.876 （W = 0.6）
Ctrl + B is Cut	Ctrl / 5 B / 1 Cut / 1	3	1.46 （W = 1） 1.168（W = 0 .8） 0.876 （W = 0.6）
Ctrl + C is Close	Ctrl / 5 C / 2 Close / 1	3	1.23 （W = 1） 0.934（W = 0 .8） 0.7 （W = 0.6）
Ctrl + ALT + D is Edit	Ctrl / 5 ALT / 2 D / 1 Edit / 1	4	1.42 （W = 1） 1.136（W = 0 .8） 0.852 （W = 0.6）
Ctrl + ALT + C is Print	Ctrl / 5 ALT / 2 C / 2 Print / 1	4	1.25 （W = 1） 1（W = 0 .8） 0.75 （W = 0.6）

Table 5: A list of facts, their corresponding concepts and each concept’s fans.

Results

The results of the simulations demonstrate that when a user attempts to retrieve multiple hotkeys from LTM in a row, the retrieval latency of a hotkey is dependent on the hotkey's history of use at times t_1, t_2, …, t_n (B_i), the user's source activation (W), and the number of reassignments of the hotkey's associated concepts (f_i). In the beginning of each round, the time required by the motor operations of a menu selection is compared to its corresponding hotkey's retrieval latency. If its retrieval latency is shorter, then the hotkey is selected to accomplish the task and the user has learned this hotkey. The subject with a source activation value of 1 learns the hotkeys for the commands "Copy", "Cut", "Print" and "Edit" after three rounds and learns the hotkey for "Close" after four rounds. The subject with a source activation value of 0.8 learns the hotkeys for the commands "Copy", "Cut" and "Print" after four rounds and learns the hotkeys for "Edit" and "Close" after 5 rounds. The subject with a source activation value of 0.6 learns the hotkeys for "Copy", "Cut" and "Print" after six rounds and learns the hotkeys for "Edit" and "Close" after seven rounds. After 50 rounds of practicing hotkeys, the retrieval latencies of the hotkeys for the three subjects are listed as:

Source Activation (W)	Copy	Cut	Close	Print	Edit
W = 1	0.187 s	0.187 s	0.266 s	0.199 s	0.258 s
W = 0.8	0.293 s	0.293 s	0.419 s	0.308 s	0.379 s
W = 0.6	0.460 s	0.460 s	0.602 s	0.477s	0.558 s

Table 6: A list of the retrieval latencies after 50 rounds of practice

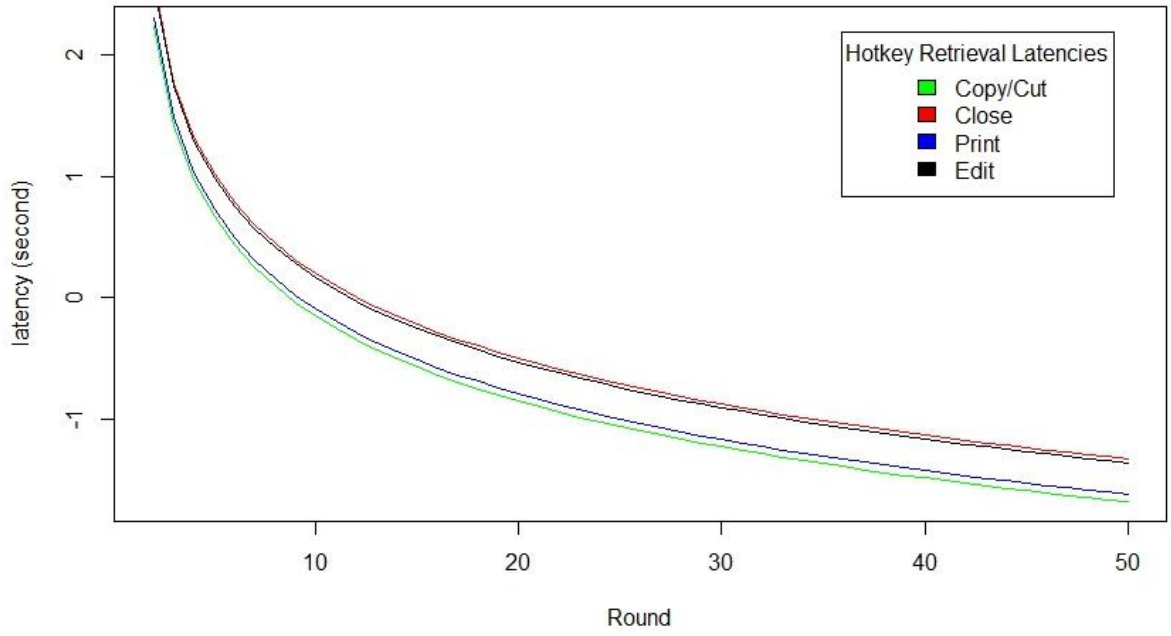

Figure 12: The retrieval latencies for the subject whose source activation value is 1.

Compared to single hotkey retrieval latencies, retrieval latencies for multiple hotkeys are not only affected by a user's source activation value and the number of practice, but

also the number of facts that a hotkey's corresponding concepts are associated with (f_i). For the same source activation value and number of practice, the hotkeys for the commands "Copy" and "Cut" have the lowest retrieval latency among the five hotkeys because they have the smallest f_i value. The hotkey for the command "Close" has the highest f_i value. Hence, it has the highest retrieval latency among the five hotkeys. The learning curves of the hotkeys for the three subjects are depicted in Figure 12 (W = 1), Figure 13 (W = 0.8) and Figure 14 (W = 0.6).

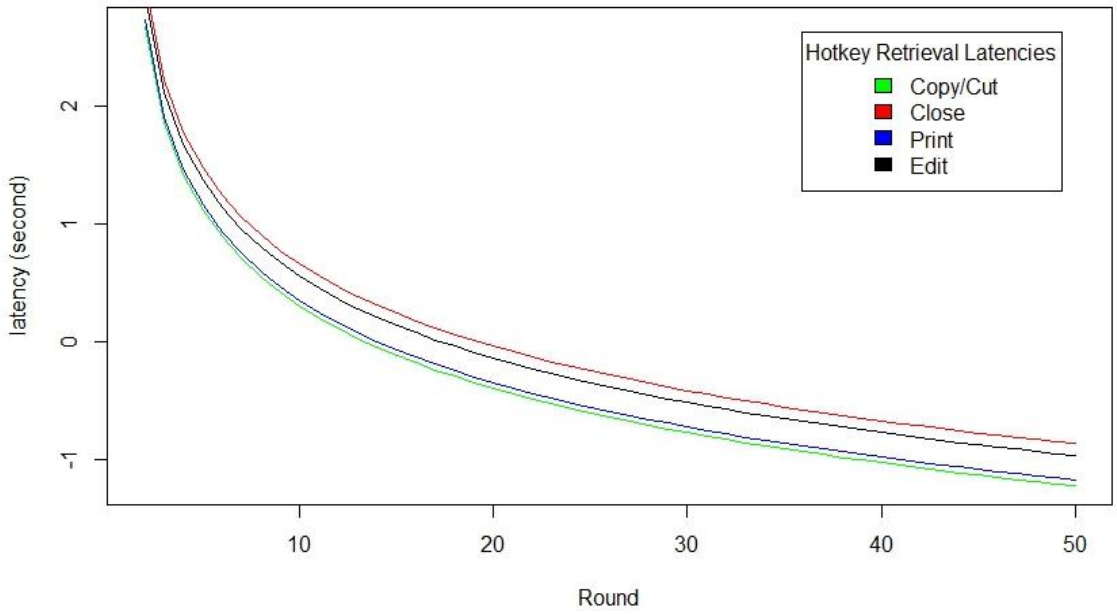

Figure 13: The retrieval latencies for the subject whose source activation value is 0.8.

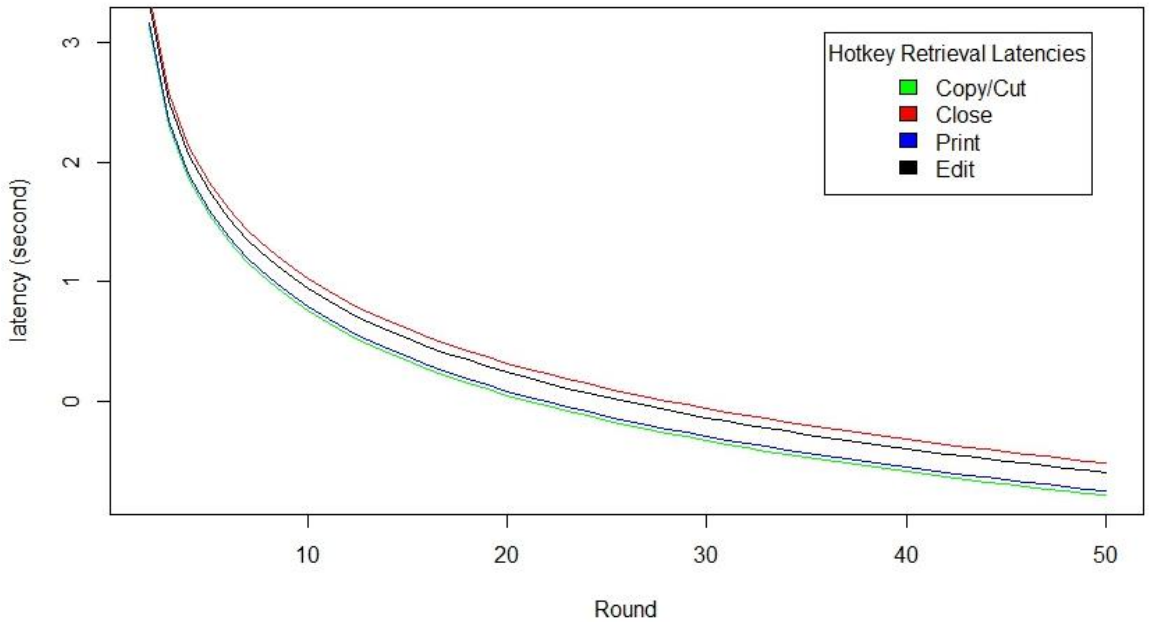

Figure 14: The retrieval latencies for the subject whose source activation value is 0.6.

5 Discussion

Compared to completing tasks via pointer-based menu selections, using hotkeys greatly relies on the successful retrievals of hotkeys. Therefore, using hotkeys is a memory-intensive strategy. Pointer-based menu selections employ an interaction-intensive strategy since it places the task knowledge on the task environments that users repeatedly interact with. Although with sufficient amount of practice, users who work with hotkeys are benefited with an advantage in task completion time, the minimum memory hypothesis suggests that the strategies that require less memory load are favored by most users. Therefore, a new hotkey interaction technique is needed to motivate users to learn hotkeys and eventually adapt hotkey use.

The primary goal of this thesis is to design an interaction technique that promotes hotkey use. In the previous chapters, a technique of displaying hotkeys through flashing keys has been developed to encourage hotkey use, and models have been constructed to predict retrieval latencies of hotkeys and simulate the hotkey learning processes for users with different working memory capacities. In this chapter, the applications of this new hotkey display technique will be discussed as well as the hotkey retrieval latency model. Additionally, this chapter also addresses the limitations of this model and future works.

5.1 Applications of Displaying Hotkey through Flashing Keys

Hotkeys are widely applied in current computer programs but they are often underused. Displaying hotkeys through flashing keys on keyboard is designed to promote hotkey use and this interactive technique presents interesting applications for various computer programs. This section introduces some examples of applying this hotkey display technology to different programs.

Command-line user interface (CLI): This is an interactive technique through which a user gives out commands to a computer program in the form of successive lines of textual input. For instance, Cisco Internetwork Operating System (IOS) is a set of software used on Cisco Systems network routers and switches. The IOS command-line user interface uses a fixed collection of multiple-word commands and contains a variety of hotkeys that correspond to certain actions. For example, when a user is typing a command, pressing the tab key automatically finishes the partial command [57]. Learning and relearning these hotkeys requires a user to check the IOS documentation. Displaying hotkeys through flashing keys can be applied to CLIs. For example, when a user is typing a seemingly long command, the tab key starts flashing to attract the user's attention and indicate that it is a hotkey. After a user issues a command, its corresponding hotkey is displayed through the flashing keys to save the user's time and effort of going through the documentation.

Emacs: This is a real-time display editor in UNIX. It normally has a menu bar at the top as well as a set of textual inputs which are used to perform certain operations. Some of the commands and textual inputs have corresponding hotkeys. The hotkeys of the commands are displayed in the parentheses after the commands. To find the hotkey for a command or a textual input, a user needs to access the menu or documentation. After applying the flashing hotkey technique, the hotkeys are displayed on the keyboard when the corresponding commands or textual inputs are issued.

Games: In many games, a hotkey enables players to map a game function into a single key or a combination of keys on the keyboard. These hotkeys are used to replace pointer-based actions for quicker and easier access to the game functions. Therefore, game players have a high incentive to use hotkeys instead of the cursor. However, displaying hotkeys on a computer screen together with heavy visual load creates visual clutter and increases users' search time for the hotkeys. Displaying hotkeys on the keyboard through flashing keys can be adapted to games to avoid visual congestion. When a player moves the cursor on a menu item, its corresponding hotkey starts flashing. Thus, the player can quickly locate the hotkeys on the keyboard. After the player moves the cursor away, the hotkey stops flashing.

Ribbons: Microsoft Office 2007 Fluent UI assembles menus and toolbars to form a ribbon in which tabbed toolbars filled with graphical widgets are grouped according to their functions and are put under different menus. Displaying a hotkey requires a user to locate the widget and move the cursor onto it for a moment. If a user clicks on the widget immediately after he moves the cursor onto it, the hotkey will not be displayed. Such a display technique fails to promote hotkey use and thus, the flashing hotkey technique should be applied to the ribbon menu. After a user clicks on a widget, the corresponding hotkey is displayed through the flashing keys. Another approach is that when a user moves the cursor on a widget, its corresponding hotkey starts to flash on the keyboard. The hotkeys stop flashing after the user moves the cursor away.

Windows 10 operating system: Windows 10 contains a set of useful functions and management tools, some of which can be accessed by their corresponding hotkeys. However, these hotkeys are not displayed in the system. For this reason, they are underused despite their usefulness and efficacy. For example, the "Run" command in Windows 10 provides direct access to a variety of functions and is often the quickest approach to launch programs, files, or documents that are stored deep in the folder hierarchy. The quickest way to access the "Run" command with a cursor is to right-click on the start menu and then select this command. Its corresponding hotkey, Windows key + R, is not displayed in the menu. The flashing hotkey display technique can be adapted to the Windows 10 operating system to promote such functions. The Windows key and letter key R start flashing when a user invokes the "Run" command through the menu selection. The keys stop flashing after the user presses them. Thus, the hotkey for the "Run" command is displayed and practiced.

Displaying hotkeys through flashing keys does not require any modification to the existing linear menus, toolbars, and ribbons. Therefore, it has a high compatibility and can easily be integrated into existing graphic user interfaces.

5.2 Applications of the Hotkey Retrieval Latency Model

According to Anderson [27], users have limited source activation and they may fail to retrieve a hotkey which is not sufficiently active. The limits on source activation vary according to each individual. The hotkey retrieval latency model accounts for the effect of source activation limits and demonstrates the hotkey learning processes of users with different memory capacities. This model can be applied to provide quantitative measures for the complicities of memory-intensive user interfaces. In the model, the parameter n is the number of elements in the focus of attention and represents the complicity of a task. For instance, when a user retrieves the command Copy's corresponding hotkey "Ctrl + C" from his Long Term Memory (LTM), n is the number of the keys (i.e. Ctrl and C) which constitute the hotkey plus one that is emerged by the fact that "Ctrl + C" corresponds to "Copy". Therefore, there are three elements in the focus of attention when a user tries to retrieve the hotkey for "Copy". A complicated task has a sufficiently large n value which increases the retrieval latency. For users who have high memory capacities, their performances degrade only when the tasks are sufficiently complicated (i.e. sufficiently high n values). However, for users with low memory capacities, their performances degrade at much lower levels of complicity. This model can be applied to estimate the memory demand of a memory-intensive user interface and whether its level of complicity is suitable for the group of users who suffer from memory impairment.

This model can also be applied to evaluate the efficiency of the hotkey vocabulary. To increase the hotkey vocabulary, many computer programs make use of multiple modifier keys (e.g. Ctrl, ALT, Windows key, and Ctrl + ALT) and reuse the same letter keys to combine with different modifier keys. For example, in Firefox, "Ctrl + N" corresponds to the command "Open New Window", and "Ctrl + ALT + N" corresponds to the command "Open New Incognito Window". The modifier key Ctrl and the letter key N are used in both hotkeys. This approach for expanding hotkey vocabulary is accompanied with a cost of increased memory demand. This model can be applied to predict the impact of a key associated with multiple hotkeys to the retrieval latencies of the hotkeys which contain this key. The model concludes that the retrieval latency of a hotkey is high if the keys which it is comprised of are associated with many other hotkeys. Therefore, hotkey vocabulary should minimize the reassignments of the same keys for constructing hotkeys in order to reduce the retrieval latencies of the hotkeys. A keyboard can be designed to contain a certain amount of extra keys to help extend hotkey vocabulary.

5.3 Limitations of the Models

The work presented in this thesis includes a model that predicts a user's selection between hotkeys and pointer-based menu selections. This model is a representation of certain perspectives of reality and therefore includes some limitations. Firstly, this model is based on an integrated theory of concurrent multitasking and assumes that moving a cursor onto a command and retrieving the command's corresponding hotkey

are processed concurrently by a user and the strategy which requires less time will be executed. The model does not account for the sequential procedure of choosing between menu selection and a hotkey, in which a user will first attempt to retrieve the hotkey, and if the hotkey's retrieval latency exceeds a certain threshold, he stops retrieving it and adapts to menu selection. However, to the best of my knowledge, no paper has researched or is able to provide a distribution for the threshold of the hotkey retrieval latency. The lack of the threshold distribution makes it unfeasible to model and simulate the sequential procedure of choosing between a hotkey and menu selection. Improving the model requires experiments in which the subjects are instructed to complete a set of tasks through either hotkeys or pointer-based menu selections. If they follow the sequential procedure, the time they spend on retrieving a hotkey before starting to move the cursor will be collected to construct a distribution for the threshold.

Another limitation of the model is that it does not account for the effects of different levels of arousal to memory. Arousal can be viewed as "a call to action with the other emotional mechanisms providing direction" [58]. A famous example of the impact of arousal is the inverted 'U' performance curve which specifies that too little arousal inhibits motivation and too much arousal inhibits concentration [59]. For the flashing hotkey display technique, the levels of arousal can be reflected through the frequencies and colors of the flashing. The hotkey retrieval latency model indicates that a user's memory of a command-hotkey paired association becomes less active since its last presentation and the rate of fading is represented by the decay parameter. Since the model does not contain any function of arousal at the time of displaying a hotkey, it does not account for the impact of the flashing frequencies and colors to the hotkey retrieval latencies.

5.4 Future Work

Future research should focus on evaluating the issues involved in integrating the technology of displaying hotkeys through flashing keys into actual computer programs. One suggestion would involve allowing users to enable and disable this technology as they desire. Another idea would involve enabling the system to identify frequently used hotkeys that have low retrieval latencies. The system disables the flashing hotkey mechanism when the associated commands of these hotkeys are invoked. Although paired-associate learning theory suggests that training on all the items at the same time produces the best results [18], being selective when applying this new hotkey display technology can make the user interface less intrusive and demanding.

Additionally, one could explore the impacts that users' emotions have in decision making and how the flashing frequencies and colors affect a user's memory of a hotkey. Prior to the hotkey display through flashing keys, users have only seen textual feedback of hotkeys. When they notice that certain keys are flashing after they have invoked a command, they will be curious to figure out the reason why the keys are flashing. Thus, spatial feedback like flashing keys may associate with users' emotions and produce better user acceptance and learning outcomes than the textual feedback.

6 Conclusion

Graphic User Interfaces (GUIs) such as linear menus and toolbar menus have become the primary interaction media between users and computer programs. This is partially due to their intuitive nature which enables users to visually search the salient graphical elements. Invoking the commands is pointer-based and GUIs support direct manipulation, which helps users to quickly learn the interfaces. However, the same features that make GUIs effective for novice users create a low ceiling for the performance of expert users. Contrarily, memory-intensive interfaces such as hotkeys are explicitly designed for experts and support high levels of performance but require extensive practice. Although the design of user interfaces for either novices or experts has been studied thoroughly, the design for user interfaces which promotes the transition from novice to expert performance has been studied less. This deficiency is reflected in the design of hotkey interaction techniques which fail to motivate users to adopt hotkey use.

Despite the advantage in task completion time which has been confirmed by various theoretical models and empirical studies, few novice users intend to learn hotkeys and employ them in their daily work. One main reason is that pointer-based access to hotkey display is unable to accentuate the existence of hotkeys and attract a user's attention. Moreover, such access introduces a performance dip which occurs in the transition to the new interface. Therefore, a new hotkey interaction technique is required to facilitate the transition from pointer-based commands to hotkeys.

This thesis proposed a new interaction technique which displays hotkeys on the keyboard. Once a user issues a command through a non-hotkey interface such as a toolbar menu or a dropdown menu, the command's corresponding hotkey is indicated on the keyboard through the flashing keys that form the hotkey. The user can press these keys to disable them from flashing. This interaction technique fulfills the design criteria by displaying hotkeys through visual stimuli. This approach can draw the users' attention to the flashing keys and thus allows important commands to have higher exposure since they are more frequently invoked. Hotkey display becomes a byproduct of issuing commands, which enables the interactions with hotkeys to avoid any non-hotkey modality. Every time a user presses the flashing keys to stop them from flashing, he is practicing using hotkeys. If a user forgets a command's corresponding hotkey, he can invoke the command in the menu, after which he can relearn the hotkey displayed through the flashing keys. Therefore, the transition from pointer-based selections to hotkeys is seamless. Moreover, this interaction technique provides incidental learning of hotkeys to novice users and helps them to break the inertia of using pointer-based menu selections. Consequently, this interaction technique of displaying hotkeys through flashing keys can facilitate the transition from pointer-based menu selections to hotkeys.

Additionally, this thesis presented a cognitive model which explains a user's decision-making process of choosing between hotkeys and pointer-based selections when the flashing hotkey technique is presented. This model is based on the concurrent

multitasking theory and asserts that a user tries to retrieve the hotkey from his Long Term Memory (LTM) while he is using pointer-based selection to complete a task. If he manages to retrieve the hotkey before the motor action of pointer-based selection is completed, he will use the hotkey for the task so that he can avoid the flashing keys. A math model was constructed with a consideration of the effects of users' working memory limitations, power law of learning, and the fan effect to accurately predict the hotkey retrieval latencies.

This thesis provides a summary of key findings from human factors literature which are crucial to the user interface designs aiming to facilitate the transition from novice to expert performance. Based on these findings, several design guidelines were formed to provide directions for designing hotkey interaction techniques. Hopefully, this thesis will provide inspiration for designing new user interfaces which facilitate expert performance as well as support ease of use.

7 Bibliography

[1] Bederson, B.B., 2004. Interfaces for staying in the flow. Ubiquity, 2004(September), pp.1-1.

[2] Remington, R.W., Yuen, H.W.H. and Pashler, H., 2016. With practice, keyboard shortcuts become faster than menu selection: A crossover interaction. Journal of Experimental Psychology: Applied, 22(1), p.95.

[3] Carroll, J.M. and Rosson, M.B., 1987. Paradox of the active user. The MIT Press.

[4] Fu, W.T. and Gray, W.D., 2004. Resolving the paradox of the active user: Stable suboptimal performance in interactive tasks. Cognitive science, 28(6), pp.901-935.

[5] Grossman, T., Dragicevic, P. and Balakrishnan, R., 2007, April. Strategies for accelerating on-line learning of hotkeys. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 1591-1600). ACM.

[6] Malacria, S., Bailly, G., Harrison, J., Cockburn, A. and Gutwin, C., 2013, April. Promoting hotkey use through rehearsal with exposehk. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 573-582). ACM.

[7] B. Shneiderman. 1987. Direct manipulation: A step beyond programming languages (excerpt). In Readings in Human-Computer Interaction: A Multidisciplinary Approach. R. Baecker and W. Buxton, eds. Morgan-Kauffman, 461–467.

[8] Jonassen, D.H. and Henning, P., 1996, July. Mental models: Knowledge in the head and knowledge in the world. In Proceedings of the 1996 international conference on Learning sciences (pp. 433-438). International Society of the Learning Sciences.

[9] Scarr, J., Cockburn, A., Gutwin, C. and Quinn, P., 2011, May. Dips and ceilings: understanding and supporting transitions to expertise in user interfaces. In Proceedings of the sigchi conference on human factors in computing systems (pp. 2741-2750). ACM.

[10] Cockburn, A., Gutwin, C., Scarr, J. and Malacria, S., 2015. Supporting novice to expert transitions in user interfaces. ACM Computing Surveys (CSUR), 47(2), p.31.

[11] Kurtenbach, G.P., 1993. The design and evaluation of marking menus (pp. 1-173). Toronto: University of Toronto.

[12] Malacria, S., Scarr, J., Cockburn, A., Gutwin, C. and Grossman, T., 2013, October. Skillometers: reflective widgets that motivate and help users to improve performance. In Proceedings of the 26th annual ACM symposium on User interface software and technology (pp. 321-330). ACM.

[13] S. K. Card, T. P. Moran, and A. Newell. 1983. The Psychology of Human- Computer Interaction. Lawrence Erlbaum Associates.

[14] E. Adar, D. S. Tan, and J. Teevan. 2013. Benevolent deception in human computer interaction. In Proceedings of the ACM Conference on Human Factors in Computing Systems. ACM, 1863–1872.

[15] Shneiderman, B. and Mayer, R., 1979. Syntactic/semantic interactions in programmer behavior: A model and experimental results. International Journal of Parallel Programming, 8(3), pp.219-238.

[16] Kurtenbach, G.P., 1993. The design and evaluation of marking menus

[17] Grossman, T., Dragicevic, P. and Balakrishnan, R., 2007, April. Strategies for accelerating on-line learning of hotkeys. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 1591-1600). ACM.

[18] Calfee, R. (1975). Associative learning. In Human Experimental Psychology. Holt, Rinehart and Winston.379-399.

[19] Kim, J.W. and Ritter, F.E., 2015. Learning, forgetting, and relearning for keystroke-and mouse-driven tasks: Relearning is important. Human–Computer Interaction, 30(1). pp.1-33.

[20] Gray, W.D., Sims, C.R., Fu, W.T. and Schoelles, M.J., 2006. The soft constraints hypothesis: a rational analysis approach to resource allocation for interactive behavior. Psychological review, 113(3), p.461

[21] Appert, C., and Zhai, S. Using strokes as command shortcuts: cognitive benefits and toolkit support. CHI ’09, ACM, 2289–2298.

[22] Rekimoto, J., Ishizawa, T., Schwesig, C., and Oba, H. Presense: interaction techniques for finger sensing input devices. UIST ’03, ACM, 203–212.

[23] Krisler, B. and Alterman, R., 2008, October. Training towards mastery: overcoming the active user paradox. In Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges (pp. 239-248). ACM.

[24] Bhavnani, S.K. and John, B.E., 2000. The strategic use of complex computer systems. Human-computer interaction, 15(2), pp.107-137.

[25] Lovett, M.C., Daily, L.Z. and Reder, L.M., 2000. A source activation theory of working memory: Cross-task prediction of performance in ACT-R. Cognitive Systems Research, 1(2), pp.99-118.

[26] Anderson, J.R., Bothell, D., Lebiere, C. and Matessa, M., 1998. An integrated theory of list memory. Journal of Memory and Language, 38(4), pp.341-380.

[27] Anderson, J.R., Reder, L.M. and Lebiere, C., 1996. Working memory: Activation limitations on retrieval. Cognitive psychology, 30(3), pp.221-256.

[28] Judith Symonds and Mehdi Khosrow-Pour. 2009. Ubiquitous and Pervasive Computing: Concepts, Methodologies, Tools, and Applications. *Information Science Reference*, Hershey, PA.

[29] Oulasvirta, A. 2015, ELEC-E7850 User Interfaces, PowerPoint presentation, Aalto University

[30] Myers, B. 1995, "User interface software tools", ACM Transactions on Computer-Human Interaction (TOCHI), vol. 2, no. 1, pp. 64-103.

[31] Laurel, B. & Laurel, B. 1990. The art of human-computer interface design, Addison-Wesley, Reading (MA).

[32] Volkswagen. 2009. Piano stairs, Available at: http://www.thefuntheory.com/piano-staircase

[33] Microsoft, 2017. *Keyboard Shortcuts for Microsoft Word 2016 for Windows*. Available at: https://support.office.com/en-us/article/Keyboard-shortcuts-for-Microsoft-Word-on-Windows-95ef89dd-7142-4b50-afb2-f762f663ceb2 [Accessed 1.3.2017]

[34] Kortum, P. & Kortum, P. 2008, HCI beyond the GUI : design for haptic, speech, olfactory, and other nontraditional interfaces, Elsevier/Morgan Kaufmann, Amsterdam.

[35] Wickens, C.D. 2008, "Multiple Resources and Mental Workload", Human factors, vol. 50, no. 3, pp. 449-455. Available at: http://journals.sagepub.com/doi/abs/10.1518/001872008X288394 [Accessed 5.3.2017]

[36] Wickens, C.D. 2002, "Multiple resources and performance prediction", Theoretical Issues in Ergonomics Science, vol. 3, no. 2, pp. 159-177. Available at: http://dx.doi.org/10.1080/14639220210123806 [Accessed 5.3.2017]

[37] Bonnie, W. Webb, N, 2010. The Handbook of Computational Linguistics and Natural Language Processing, John Wiley & Sons

[38] Bigelow J, Poremba A, 2014. Achilles' Ear? Inferior Human Short-Term and Recognition Memory in the Auditory Modality. Available at: doi:10.1371/journal.pone.0089914 [Accessed 6.3.2017]

[39] Newell, A., and Rosenbloom, P. S. The soar papers (vol.1). MIT Press, Cambridge, MA, USA, 1993, ch. Mechanisms of skill acquisition and the law of practice, 81–135.

[40] Bailly, G. 2014, "Model of visual search and selection time in linear menus". Available at: http://dl.acm.org/citation.cfm?id=2557093 [Accessed 9.3.2017].

[41] Fitts, P.M. 1954. The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology, 47, 381-391. Available at : http://psycnet.apa.org/journals/xge/47/6/381/

[42] Chapuis, O. and Dragicevic, P. (2011) Effects of motor scale, visual scale, and quantization on small target acquisition difficulty. ACM Trans. Comput.-Hum. Interact.

[43] Bi, X. 2013, "FFitts law: modeling finger touch with fitts' law". Available at: http://dl.acm.org/citation.cfm?id=2466180 [Accessed 9.3.2017].

[44] Anderson, J.R. 2004, "An Integrated Theory of the Mind", Psychological review, vol. 111, no. 4, pp. 1036-1060. Available at: http://www.jakubdotlacil.com/tutorials/integrated%20theory%20of%20mind.pdf [Accessed 14.3.2017]

[45] Anderson, J.R. 2007. How can the human mind occur in the physical universe? Oxford University Press, New York.

[46] Wilson, M. (2002). Six views of embodied cognition. Psychonomic Bulletin & Review, 9(4), 625–636.Availbale at: https://people.ucsc.edu/~mlwilson/publications/Embodied_Cog_PBR.pdf [Accessed 9.3.2017].

[47] Salvucci, D. D. 2008. Threaded Cognition: An Integrated Theory of Concurrent Multitasking. Psychological Review, 115(1), pp. 101-130.

[48] Norman, D. A., & Shallice, T. (1986). Attention to action: Willed and automatic control of behavior. In R. J. Davidson, G. E. Schwartz, & D. Shapiro (Eds.), Consciousness and self-regulation (pp. 1–18). New York: Plenum

[49] Cooper, R., & Shallice, T. (2000). Contention scheduling and the control of routine activities. Cognitive Neuropsychology, 17, 297–338.

[50] Howes, A., Lewis, R.L. and Vera, A., 2009. Rational adaptation under task and processing constraints: implications for testing theories of cognition and action. Psychological review, 116(4), p.717.

[51] Card, S. 1980. The keystroke-level model for user performance time with interactive systems. Communications of the ACM, 23(7), pp. 396-410.

[52] Olson, J. R., & Olson, G. M. 1990. The growth of cognitive modeling in human-computer interaction since GOMS. Human-Computer Interaction, 5, 221-265.

[53] Anderson, J. R. 1993. Rules of the mind. Hillsdale, NJ: Erlbaum.

[54] Anderson, J. R., & Reder, L. M. 1999. The fan effect: New results and new theories. Journal of Experimental Psychology General, 128, 186-197.

[55] Lovett, M. C., Daily, L. Z., & Reder, L. M. 2000. A source activation theory of working memory: Cross-task prediction of performance in ACT-R. Cognitive Systems Research, 1(2), 99-118.

[56] Serna, A., Pigot, H., & Rialle, V. 2007. Modeling the progression of Alzheimer's disease for cognitive assistance in smart homes. User Modeling and User-Adapted Interaction, 17(4), 415-438.

[57] Sequeira, A., 2013. Interconnecting Cisco Network Devices, Part 1 (ICND1) Foundation Learning Guide. Cisco Press.

[58] Chown, E., Cochran, R.E. and Lee, F.J., 2006, January. Modeling Emotion: Arousal's Impact on Memory. In Proceedings of the Cognitive Science Society (Vol. 28, No. 28).

[59] Yerkes, R. M. & Dodson, J. D. 1908. The relation ofstrength of stimulus to rapidity of habit formation,Journal of Comparative Neurology and Psychology, 18, 459–482.